Synthesis and evaluation of engineering processes for the development of airborne electronic equipment

DA Viljoen
20264763

Thesis submitted for the degree *Philosophiae Doctor* in Development and Management Engineering at the Potchefstroom Campus of the North-West University

Promoter: Prof JEW Holm

October 2016

Acknowledgements

I wish to acknowledge the contributions made by all my colleagues at Denel Aviation, the South African Air Force, Armscor, and the other companies who co-operated with Denel Aviation on the projects described in the thesis, notably M-TEK, Saab, Sagem and CMC Esterline, in terms of refining and improving my understanding the concepts described in this thesis.

I also want to acknowledge the role of distinguished mentors who assisted me in understanding the topics addressed in this study, notably Professor Johann Kruger, Professor Ad Sparrius and Doctor Jerry Lake.

Thank you to Wes Lindenberg who provided the photos.

A very special thank you to my son, Daniël, who undertook the immense task of proofreading the dissertation.

I wish to thank my employer, Denel Aviation, for creating the opportunity to complete this study.

I am greatly indebted to my study leader, Professor Johann Holm, for his guidance, enthusiasm, patience and motivation.

Finally, thank you to my wife Thelma and my family for their love, support and encouragement through many years.

Vir Albertus.

In herinnering.

Abstract

Electronic equipment installed on aircraft has to meet with airworthiness requirements and standards. Airworthiness, i.e. confidence that the equipment is suitable for its intended functions and safe to operate, is established by scrutiny and the meticulous control of the data associated with the system design and associated engineering processes, manufacturing and installation processes, operating instructions and procedures, and the maintenance aspects of the equipment. All deviations, including upgrades and modifications to any of these aspects, are subjected to prudent controls. A "paper trail" of technical decisions, implementation details, validation and verification results and evidence of adequate control of these must be established and maintained throughout the operational life of the equipment for each operational version of the affected systems. In particular, equipment containing embedded software is generally highly complex, making this effort of managing the airworthiness related information a demanding undertaking.

A significant number of sources, e.g. aerospace industry recommended practises, system and software engineering standards, and other relevant publications, contribute to the set of references against which airworthy products are developed. The information contained in these sources is of a generalised nature and typically represents particular points of view, depending on the source. The formulated problem was therefore to derive a process description specific to the development of airborne electronic equipment, which complies with this collection of requirements efficiently by detailing a comprehensive and consolidated methodology.

In order to provide a scientific grounding for the study, techniques associated with the methodology of Design Science Research (DSR) were applied. The general principle of DSR is that an innovative, purposeful artefact is created for a specified and adequately formulated problem domain and a mechanism to provide a solution is proposed. The artefact, which represents the aforementioned solution, must be systematically evaluated and the result must be integrated in practise. Most accumulated scientific knowledge can be classified as descriptive knowledge, which contains descriptions of phenomena and interrelationships between phenomena. DSR is a method for establishing prescriptive knowledge, which entails constructs, models, methods, instantiations and design theories. This study is concerned with knowledge of a method that meets with the definition of prescriptive knowledge.

The artefact, within the context of the DSR method, is a definition of a unified process for the development of airborne electronic equipment. This description identifies the necessary and sufficient set of activities required to develop and support such a system and the accompanying documentation efficiently, within the specific constraints applicable to Denel Aviation. This process description is based on applicable aerospace industry recommended practises and other prevalent systems engineering references, as well as on practical experience in the development of such systems. In particular, experience from the development of an automatic flight control system for an attack helicopter, and the integration of a modern navigation system on a medium transport helicopter, were applied in the formulation of this process description.

The process description is implemented by Denel Aviation as a company standard for the development and airworthiness assurance of airborne electronic equipment, and is used for the detailed planning of new projects. The process of formally approving the process description for use by Denel Aviation served to validate the outcome of the study in terms of DSR principles.

Keywords: Process, airworthiness, Design Science Research, airborne electronic equipment.

Opsomming

Elektroniese toerusting wat op vliegtuie gebruik word moet aan lugwaardigheidsvereistes en standaarde voldoen. Lugwaardigheid van 'n stelsel, d.i. die vestiging van vertroue dat die stelsel geskik is om die funksies uit te voer waarvoor dit beplan is en dat dit veilig is om te gebruik, berus o.a. op die deeglike ondersoek van die data wat die ontwerp van die stelsel omskryf en die versekering van die korrektheid daarvan. Enige veranderinge aan hierdie data word noukeurig beheer. Alle tegniese besluite, sowel as die resultate van analises en toetse om die korrektheid van die stelsel mee te verifieer, moet gedokumenteer word vir elke weergawe van die stelsel en die dokumentasie moet beskikbaar wees vir ontledings solank as wat die stelsel operasioneel aangewend word. Elektroniese stelsels wat ingebedde sagteware bevat is besonder kompleks van aard, wat die taak om data wat met lugwaardigheid gepaardgaan te ontwikkel en te bestuur uitdagend maak.

'n Beduidende aantal bronne, byvoorbeeld lugwaardigheidsvoorskrifte, standaarde vir die ontwikkeling van stelsels en sagteware, en nog ander, vorm deel van die stel verwysings waarteen lugwaardige stelsels ontwerp word. Die inligting wat in hierdie dokumente vervat is, is gewoonlik in die vorm van breë veralgemeende beskrywings, en verskillende bronne lê klem op verskillende aspekte, wat tipies die doelwitte van die organisasie wat die bron publiseer ondersteun. Die navorsingsprobleem was gevolglik om 'n proses vir die ontwikkeling van lugwaardige elektroniese te formuleer wat doeltreffend aan al die toepaslike voorskrifte voldoen, deur middel van 'n omvattende maar gekonsolideerde beskrywing.

Die studie het die metodologie van ontwerpswetenskapnavorsing (Design Science Research of "DSR") toegepas. In die DSR-proses word 'n innoverende en doelmatige artefak geskep vir 'n gespesifiseerde en behoorlik geformuleerde probleemomgewing, en 'n meganisme om 'n oplossing te voorsien word voorgestel. Die artefak wat die voorgestelde oplossing implementeer moet stelselmatig geëvalueer word en die resultaat moet in die praktyk toegepas word. Die meerderheid van wetenskaplike kennis kan as beskrywende kennis geklassifiseer word, waar verskynsels en verwantskappe tussen verskynsels beskryf word. DSR is 'n metodiek vir die vestiging van voorskriftelike kennis, wat modelle, metodes en ontwerpsteorieë behels. Hierdie studie het die ontwikkeling van kennis van 'n metode behels, en voldoen dus aan die definisie van voorskriftelike kennis.

'n Definisie van 'n gekonsolideerde proses vir die ontwikkeling van elektroniese stelsels vir gebruik aan-boord vliegtuie is as 'n uitkomste van die studie ontwikkel. Hierdie beskrywing identifiseer 'n noodsaaklike en voldoende stel aktiwiteite om so 'n stelsel, en al die gepaardgaande dokumentasie, doeltreffend te ontwikkel binne die spesifieke beperkinge wat op Denel Aviation van toepassing is. Hierdie prosesbeskrywing is gebaseer op voorskrifte wat op die lugvaartbedryf en op ander stelselingenieursweseverwysings van toepassing is, sowel as op praktiese ondervinding in die ontwikkeling van sodanige stelsels. Lesse wat uit die ontwikkeling van 'n outomatiese vlugbeheerstelsl vir 'n aanvalshelikopter geleer is, en die integrasie van 'n moderne navigasiestelsel op 'n veeldoelige mediumgrootte helikopter, is spesifiek toegepas in die formulering van die prosesbeskrywing.

Hierdie proses is deur Denel Aviation geïmplementeer as 'n interne maatskappystandaard vir die ontwikkeling en lugwaardigheidsversekering van elektroniese toerusting. Die prosesbeskrywing word ook gebruik in die gedetailleerde beplanning van nuwe projekte. Die formele goedkeuringsproses van die standaard dien o.a. as validasie van die prosesbeskrywing.

Sleutelwoorde: prosesbeskrywing, lugwaardigheid, ontwerpswetenskapnavorsing, electroniese stelsels.

Table of Contents

List of Figures

List of Tables

Abbreviations

ABL	Allocated Baseline
ADM	Advanced Development Model
AC	Advisory Circular (FAR related)
AESDP	Airborne Electronic Systems Development Process
AFCS	Automatic Flight Control System
AIU	Avionics Interface Unit
ANSI	American National Standards Institute
AP	Autopilot
ARINC	Aeronautical Radio Incorporated
Armscor	Armaments Corporation of South Africa
ARP	Aerospace Recommended Practice
AS	Aerospace Standard
ASIC	Application-specific Integrated Circuit
ATP	Acceptance Test Procedure
BKCASE	Body of Knowledge and Curriculum to Advance Systems Engineering
CAA	Civil Aviation Authority
CAD	Computer Aided Design
CC1	Control Category 1
CC2	Control Category 2

CCB	Change Control Board
CDR	Critical Design Review
CFT	Certificate for Flight Trials
CI	Configuration Item
CM	Configuration Management
CMP	Configuration Management Plan
CMMI	Capability Maturity Model Integration
CoD	Certificate of Design
COTS	Commercial off the Shelf
CRA	Commissioning Readiness Audit
CRT	Cathode Ray Tube
CSCI	Computer Software Configuration Item
DAFA	Directorate: Air Force Acquisitions
DDBD	Database Design Description
DDP	Declaration of Design and Performance
DEF STAN	Defence Standard
DID	Data Item Description
DoD	Department of Defence (USA)
DSI	Directorate System Integrity
DSR	Design Science Research
DTU	Data Transfer Unit
EASA	European Aviation Safety Agency

ECP	Engineering Change Proposal
EDM	Evaluation Development Model
EIA	Electronic Industries Alliance
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
EUROCAE	European Organisation for Civil Aviation Equipment
FAIT	Fabrication, Assembly, Integration, and Test
FAA	Federal Aviation Authority
FAR	Federal Aviation Regulation
FADEC	Full Authority Digital Engine Control
FAR	Federal Aviation Regulation
FBL	Functional Baseline
FCA	Functional Configuration Audit
FHA	Functional Hazard Assessment
FMEA	Failure Mode and Effect Analysis
FPGA	Field-Programmable Gate Array
FTA	Fault Tree Analysis
FTG	Flight Test Group
FTI	Flight Test Instrumentation
FTR	Flight Test Request
GLCM	Generalised Lifecycle Model
GPS	Global Positioning Satellite / System

HF	High Frequency
HMI	Human - Machine Interface
HTS	Hazard Tracking System
HWCI	Hardware Configuration Item
ICAM	Integrated Computer Aided Manufacturing
ICD	Interface Control Document
IDD	Interface Design Description
IDEF0	ICAM definition for functional modelling
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
IID	Incremental and Iterative Development
INCOSE	International Council on Systems Engineering
IOBL	Initial Operational Baseline
IRS	Interface Requirements Specification
ISO	International Standards Organisation
JAR	Joint Aviation Regulation
JAS	Joint Air Strike
LCD	Liquid Crystal Display
LRU	Line Replaceable Unit
MAB	Military Airworthiness Board
MBL	Manufacturing Baseline
MCDU	Mode Control and Display Unit

MFD	Multi-Function Display
MFDP	Multi-Function Display Processor
MIL-STD	Military Standard
MoC	Means of Compliance
MoD	Ministry of Defence (UK)
MRI	Master Record Index
OBL	Operational Baseline
OCD	Operational Concept Definition
OSHA	Occupational Safety and Health Act
OT&E	Operational Test and Evaluation
PBL	Product Baseline
PCA	Physical Configuration Audit
PCMCIA	Personal Computer Memory Card International Association
PDR	Preliminary Design Review
PHAC	Plan for Hardware Aspects of Certification
PLD	Programmable Logic Device
PLM	Product Lifecycle Management
PM	Production Model
PM	Program or Project Management or Manager
PPM	Pre-Production Model
PRACAS	Problem Reporting and Corrective Action System
PRR	Production Readiness Review

PSAC	Plan for Software Aspects of Certification
PSSA	Preliminary System Safety Assessment
QA	Quality Assurance
QAP	Quality Assurance Plan
QAR	Quality Assurance Representative
QC	Quality Control
QMS	Quality Management System
RAM	Reliability, Availability, Maintainability
RBL	Requirements Baseline
Req	Requirement
RF	Radio Frequency
RFC	Request for Change
RP	Recommended Practise
RSA	Republic of South Africa
RTCA	Radio Technical Commission for Aeronautics
SAAF	South African Air Force
SAE	Society of Automotive Engineers
SCI	Software Configuration Index
SDD	Software Design Description
SDF	Software Development File
SE	Systems Engineering
SEBoK	Systems Engineering Body of Knowledge

SEMP	Systems Engineering Management Plan
SEP	Systems Engineering Plan
SLCI	Software Lifecycle Configuration Index
SPS	Software Product Specification
SRS	Software Requirements Specification
SRU	Shop Replaceable Unit
SSA	System Safety Assessment
SSDD	System/Subsystem Design Description
SDP	Software Development Plan
SIP	Software Installation Plan
SSS	System/Subsystem Specification
STD	Standard
STrP	Software Transition Plan
STP	Software Test Plan
STR	Software Test Report
SVD	Software Version Description
SW	Software
SwCI	Software Criticality Index
TEMP	Test and Evaluation Master Plan
TC	Type Certificate
TSO	Technical Standard Order
UHF	Ultra High Frequency

UK	United Kingdom
UML	Unified Modelling Language
URS	Users Requirements Specification
URR	User Requirements Review
USA	United States of America
USB	Universal Serial Bus
VHF	Very High Frequency
vs.	versus
XDM	Exploratory Development Model

Chapter 1
Introduction

Denel Aviation develops and modifies electronic systems and integrates them onto aircraft as one of the elements of its business. This study describes the derivation of a formalised process for the efficient and effective development of such systems, particularly in the context of the development of systems for use by South African military establishments. Notably, the introduction of embedded digital computers in airborne applications significantly increased the complexity of these systems, thereby compounding the objective to develop these systems to operate correctly and safely.

1.1 Rationale

The purpose of the study was to define and validate an authoritative process for the development of electronic equipment containing embedded software that meets with the requirements and constraints applicable to airborne systems. The study analysed shortcomings associated with the development of electronic systems on board military aircraft in the South African context and identified shortcomings in existing practises, implemented and evaluated improvements, and proposed further enhancements. A further important objective was to encapsulate the aforementioned activities in an academic framework, with the intention to perform an abstraction from observations, that is, to generalise as much as possible.

1.2 Scope of the study

In order to refine the scope of the study described in this dissertation, a distinction is made between platform specific systems and platform independent systems:

- **Platform dependent systems** are integrated into a particular airframe type and avionics architecture, according to a given set of user requirements and dependent on specific aircraft and avionics characteristics (i.e. customised

equipment developed against a specific agreement). This class of systems includes mission computers, automatic flight control systems, health monitoring systems and avionics integration systems. The system forms part of the type certificate of an aircraft and certification testing of the system takes place in the context of the integrated air vehicle.

- **Platform independent systems** are designed against a market-driven set of generic requirements and can typically be installed in a number of different aircraft types where the functionality is not dependent on the aircraft type onto which it is integrated. Examples of such generic systems are attitude and heading reference systems, cockpit displays and radio transceivers.

Due to the differences in establishing system requirements and in the airworthiness approval processes, different methodologies are applied during the development of these two types of systems. Denel Aviation develops platform specific systems, therefore the processes for the development of these types of systems are the focus of the study.

1.3 Problem statement

The processes involved in the development of airborne electronic equipment are extensive and many interrelationships exist between these processes. Specifications for outcomes to be accomplished by these processes originate from a number of diverse sources and, consequently, there appears to be a lack of consensus on specifics with regards to systems development. In establishing a design for a development process for a specific purpose, commonality of purpose must be derived from these sources, while the duplication of effort must be minimised and eliminated. At the same time, it must be ensured that the resulting work breakdown structure is sufficiently comprehensive in order to meet with all prerequisites for product certification, while meeting productivity criteria and minimising development risks. The development process must be synthesised to meet with the specific objectives of the type of system being developed, taking cognisance of the specific environment where it is being developed.

1.4 Research purpose / goal

The purpose of the research can thus be formulated as follows:

Synthesise and validate a process to ensure cost effective development of platform specific airborne electronic equipment in the South African industrial and military environment.

1.5 Dissertation outline

The content of this dissertation is organised as set out below.

- The research methodology and processes are described in Chapter 2.
- The research problem is formulated and validated in Chapter 3. Essential characteristics of airborne electronic systems are described for background purposes. Both the history of the development of airborne electronic equipment and the evolution of the associated engineering processes at Denel Aviation are recounted. A retrospective assessment of two projects concerning the development of airborne electronic equipment undertaken by Denel Aviation is presented, and the research challenges emanating from these assessments are formulated.
- A review of systems engineering notions and models as they were formulated in classical systems engineering standards, as well as is in contemporary systems engineering publications, including quality and configuration management standards and airworthiness recommended practises, is presented in Chapter 4. The literature study also includes a review of lifecycle models described in academic literature. The literature study serves to render further validation of the research challenges, and indicated potential solutions to these requirements. The synthesis of a generalised lifecycle model appropriate to the development of airborne electronic equipment is presented at the end of the chapter.
- The detailed design of the development process, based on the generalised lifecycle model, is described in Chapter 5.
- The validation of the elements of the research process is presented in Chapter 6.
- Conclusions are summarised in Chapter 7.

1.6 Summary

The topic of the dissertation, namely the development and validation of a process for the development of airborne electronic equipment that utilises embedded software within the South African context, was introduced. The study objectives were detailed and the scope of the study was defined. A definitive problem statement was introduced and an outline of the dissertation was presented.

Chapter 2
Research approach

2.1 Introduction

In their book "Practical Research and Evaluation", Dahlberg and McCaig [1] commence their introductory paragraph with the statement: "In our knowledge society, there is an expectation that practise should be evidence based". It follows that practitioners in a given area of activity should base their actions on information gained from objective inquiry into the processes and methodologies employed in said area of interest.

In the light of this notion, the aim of the research described in this dissertation is to obtain an understanding of the engineering processes employed in support of the development of a certain class of airborne electronic equipment. Such an understanding will provide practitioners with sufficient information to enable the efficient execution of development programs based on corroborated proof of the concepts underlying the technical processes.

The study described in this dissertation applied techniques associated with the methodology of Design Science Research (DSR). DSR aims to solve real-world problems by performing an abstraction of the real-world problem and producing an artefact that represents a solution to that specific problem. The artefact that represents the solution must be systematically evaluated and the result must be integrated in practise.

Whereas most accumulated scientific knowledge can be classified as descriptive knowledge, which contains descriptions of phenomena and interrelationships between phenomena, DSR is a method for establishing prescriptive knowledge. Gregor and Hevner [2] summarise prescriptive knowledge as knowledge concerning artefacts designed by humans to improve the natural world. The prescriptive knowledge base contains constructs, models, methods, instantiations and design theories. This study is concerned with knowledge of a method that is an element of the prescriptive knowledge base.

2.2 Research process classification

The research process followed in this study can be classified as applied, empirical and qualitative (Dahlberg & McCaig [1]). While basic research aims at the development of

knowledge for the sake of knowledge only, applied research aims to obtain knowledge with the objective of using the knowledge for a commercial or practical purpose. As noted earlier, the study is also empirical, as knowledge is improved by this study based on observations of reality in the form of abstractions.

Dahlberg & McCaig [1] describe the differences between qualitative and quantitative research as follows:

> "Qualitative descriptions are stated in verbal terms, and differ in the kind of descriptions, whereas quantitative descriptions provide numerical quantifications and differ in numbers and degree. Qualitative research typically covers a few cases; quantitative research involves many cases. Qualitative methods utilise in-depth interviews while quantitative methods employ questionnaire surveys, both types use observations and content analysis. From an ontological viewpoint, the reality described by results of qualitative investigations is established by the perceptions of the investigator whereas quantitative research yields a reality independent of perceptions. Epistemologically, qualitative research produces subjective knowledge and bias cannot be avoided; quantitative research collects objective data. It is more problematic to generalise qualitative findings to a general population than what is possible with quantitative results."

As the research is of a qualitative nature, the research process was inductive rather than deductive. Empirical data was obtained during the investigation, and this data led to more questions that could be posed to increase the understanding of the problem at large and contributed to the development of a theoretical model that found application in the derived process description. Therefore, inductive reasoning is iterative in nature, as reflected in this research.

2.3 Design science research (DSR)

The DSR methodology originated with the purpose of establishing knowledge and solving problems in the domain of information systems research. This concept is stretched with respect to this study in order to incorporate a specific type of system that relies on software for its operation. Hevner *et al* [3] state that the fundamental principle "... is that knowledge and understanding of a design problem and its solution are acquired in the building and application of an artefact". In the instance of this study, the artefact is a comprehensive Development Process Description, which was subjected to evaluation and improvement. This process definition is based on a basic generalised lifecycle model that was abstracted from prevalent international systems

engineering and software development references. This approach deviates from the original domain by focussing on the synthesis of a development process model as opposed to research in information technology, but the principles of design science research are suitably universal and are applicable to research of this nature.

Hevner *et al* [3] present seven guidelines for design science research in information systems, which are all applicable to this research.

> Guidelines 1 to 4 can be summarised and paraphrased as follows: An innovative, purposeful artefact is created for a specified problem domain and is systematically evaluated. The artefact must be innovative, either solving an as yet unsolved problem or solving a known problem in a more effective or efficient manner.
>
> Guideline 5 states that "the artefact itself must be rigorously defined, formally represented, coherent, and internally consistent".
>
> Guideline 6 requires that the problem space is formulated and that a mechanism is posed to find an effective solution.
>
> Guideline 7 states that the results of the research must be communicated effectively and integrated in practise.

Hevner [4] provides an elegant summary of the DSR process. The DSR process is described as a "three cycle view", shown in Figure 1, and consists of the Design Cycle, the Relevance Cycle and the Rigour Cycle. In this view, the design cycle is the most essential element of the DSR process, where design artefacts and processes are developed and evaluated.

The DSR process associates with the environment by means of the relevance cycle. The relevance cycle describes the determination of the research requirements from the application domain, and the eventual operational validation (field-testing) of the design artefact. In other words, the need for the research emanates from the environment where the artefact is to be deployed, and the utility of the artefact is eventually evaluated in this environment.

Hevner [4] indicates that the rigour cycle associates the design cycle with the knowledge base of scientific theories and engineering methods. Hevner [4] states that:

> "The rigour cycle provides past knowledge to the research project to ensure its innovation. It is contingent on the researchers to thoroughly research and reference the knowledge base in order to guarantee that the design produced

are research contributions and not routine designs based on designs based on the application of well-known processes."

Hevner [4] argues that, in addition to descriptive theories, design science can be grounded in aspects such as existing artefacts and analogues/metaphors. Hevner [4] points out that the knowledge base contains two types of additional knowledge, i.e. experience and expertise that define the state-of-the-art in the application domain of the research, and the existing artefacts and processes found in the application domain.

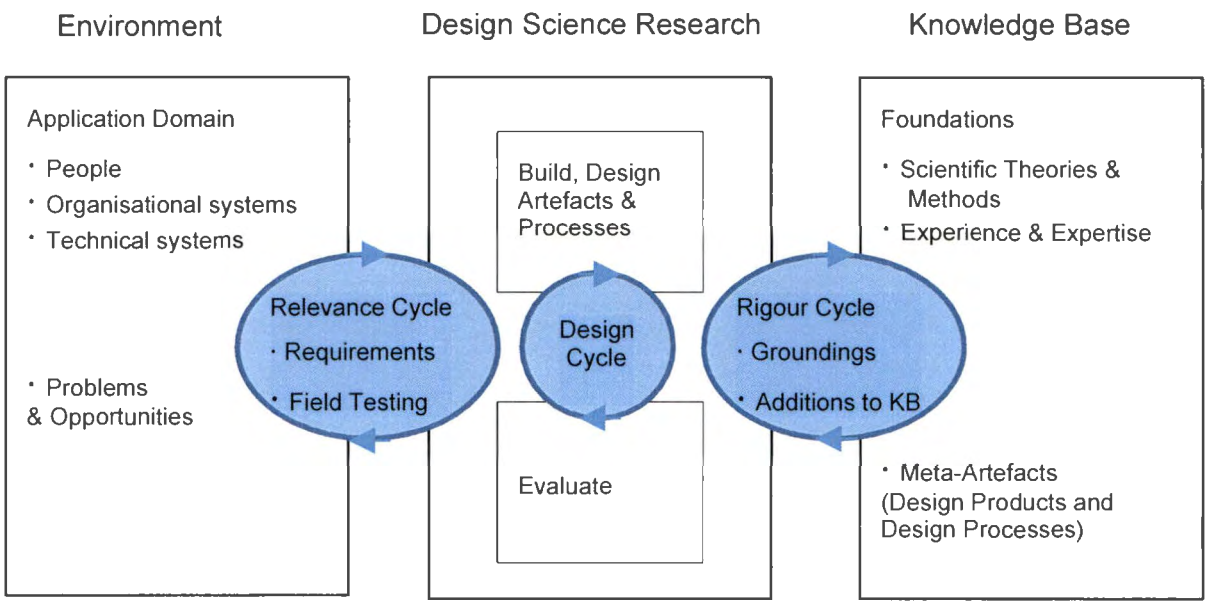

Figure 1: The design science research cycles (Hevner [1])

2.4 Application of DSR to this study

The research requirements element of the relevance cycle regarding this study is presented in Chapter 3 of this dissertation, formulated in the form of research challenges. The research challenges were determined by way of an assessment of engineering processes applied within the organisation, as well as by means of retrospective assessments of two development projects executed by Denel Aviation.

The element of establishing the groundings for the development of a design as a component of the rigour cycle is achieved in Chapter 4. The domain knowledge base is researched to identify theories and methods for the establishment and evaluation of the development process (the DSR "artefact"). In particular, aspects influencing engineering concepts described in the following areas of interest are researched, with reference to the research challenges identified in Chapter 3:

- Classical systems engineering approaches;

- Contemporary systems engineering standards;
- Quality management considerations;
- Configuration management considerations;
- Airworthiness recommended practises;
- Lifecycle models described in academic literature.

The design cycle is described in Chapter 5. A lifecycle model for the development of airborne electronic equipment was derived, based on findings of the literature study presented in Chapter 4. This model served as a framework for the design of a comprehensive process definition where the lifecycle model is used recursively and iteratively. The comprehensive process definition was subjected to a peer evaluation for validation purposes.

The additions to the knowledge base are described in Chapter 7.

2.5 Research knowledge contribution

A fundamental issue stated by Gregor and Hevner [2], is that nothing is really "new" as everything is made of something else or builds on some previous idea(s). To determine when something is really novel or has a significant impact to the knowledge base, a DSR knowledge contribution framework was presented by Gregor and Hevner [2]. This framework classifies different types and levels of research contributions according to starting points from the research in terms of problem maturity and solution maturity. The matrix representing this framework is shown below in Figure 2.

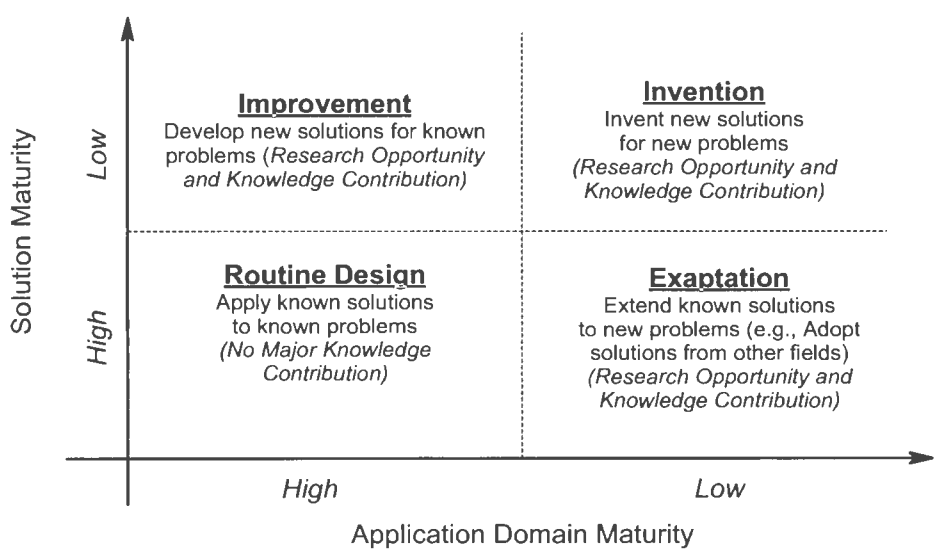

Figure 2: DSR knowledge contribution framework [1]

With reference to Figure 2, the knowledge contribution for this study could be classified to fall in the top left quadrant of the matrix, i.e. a new (improved) solution to a known problem is provided by this research. According to Gregor and Hevner [2], the key objective of the research type in this quadrant is the demonstration that the improved solution adds to the knowledge base. In order to surmount this research objective, it is shown that the process definition that was the primary outcome of this research is implemented in practise and that it solved the shortcomings that are identified in Chapter 3.

2.6 Conclusion

This chapter presented the research methodology applied in this dissertation. The DSR framework was explained in general, and the approach to this research was aligned with the DSR framework. The knowledge contribution for this research was classified to be "a new improved solution to a known problem".

Chapter 3
Validation of the research problem

3.1 Overview

The objective of this chapter is to formulate and validate the research challenges, which form an integral part of the relevance cycle of the DSR method. The environment in which the development of airborne electronic equipment takes place is described, and problems encountered during the development of these systems are described as well.

A background to the development of airborne electronic equipment is presented first. A summary of the evolution of engineering processes within Denel Aviation related to the development of systems containing embedded software is presented in the next section, followed by a retrospective assessment of two major engineering programs that have been executed by Denel Aviation. Finally, this chapter is concluded with the formulation of the research challenges.

The retrospective assessments presented in this chapter serve to substantiate specific aspects of the central research problem addressed in this dissertation and to impart the uniqueness of the research problem.

Note: In order to provide inferences and to simplify reading of this thesis, Italic font (cursive) text is used to present deductions from the retrospective assessments and literature study, or to show relevance of literature to this research.

3.2 Background

3.2.1 Airborne electronic equipment

Electronic devices and systems have been used on board aircraft since the dawn of controllable powered flight and are employed in a multitude of roles to assist the crew in executing their duties and to improve the mission capability of the aircraft in which these devices and systems are installed.

Until the advent of practical embedded computers (during the early 1980s), electronic systems on board aircraft usually operated independently from one another. A system

with a particular function was essentially self-contained and comprised dedicated input devices or sensors, output devices or actuators, analogue and Boolean logic signal processors, and human-machine interface (HMI) devices. A generalised diagram of a typical airborne electronic system is shown in Figure 3.

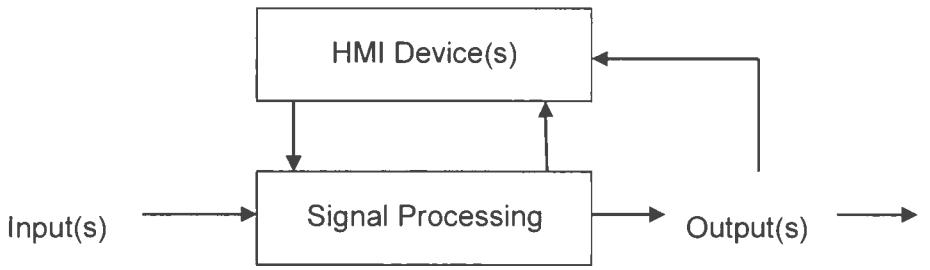

Figure 3: Stand-alone Airborne Electronic System

The inputs could e.g. be radio frequency (RF) signals from an antenna, in the case of a radio transceiver, rates and attitudes from gyroscopes as used by flight instruments and autopilots or temperature and air pressures used by a bombing computer. The outputs could typically be audio signals, flight control commands to servos or driver signals to HMI devices. The HMI devices could be indicator lamps, dials, switches, seven segment displays, and/or control knobs. The signal processing block would perform the required functions to convert the information from the inputs and input HMI devices to output voltages or currents and output display indications.

The development of embedded digital computers introduced intelligence and the capability to share data and other resources (e.g. HMI displays) between systems, thereby enhancing functional capabilities and improving the effectiveness of aircrews, specifically under high workload conditions. This sharing of data and resources has led to the notion of integrated avionics systems. A generalised schematic diagram for an integrated avionics system is shown in Figure 4.

Flight data sensors include equipment for the measurement of aircraft attitudes, rates and accelerations, air pressures and temperature. Navigation sensors provide information about the geographical position of the aircraft. Health monitoring sensors include pressure sensors, thermocouples, flow rate meters, and other transducers mounted on the engines, transmission devices, hydraulic, fuel and electrical systems.

Flight control actuators typically employ servo devices to drive flight control surfaces or rotor system controls. Radio transceivers for airborne use cover High Frequency (HF), Very High Frequency (VHF) and Ultra High Frequency (UHF) ranges. The data processors are typically single task embedded computers. The HMI devices include

multi-function displays, keyboards and a variety of switches. The most widely used data transfer protocols are those defined in ARINC 429[5] and MIL STD 1553B [6], although a number of other protocols, such as RS-422 and Ethernet are also used.

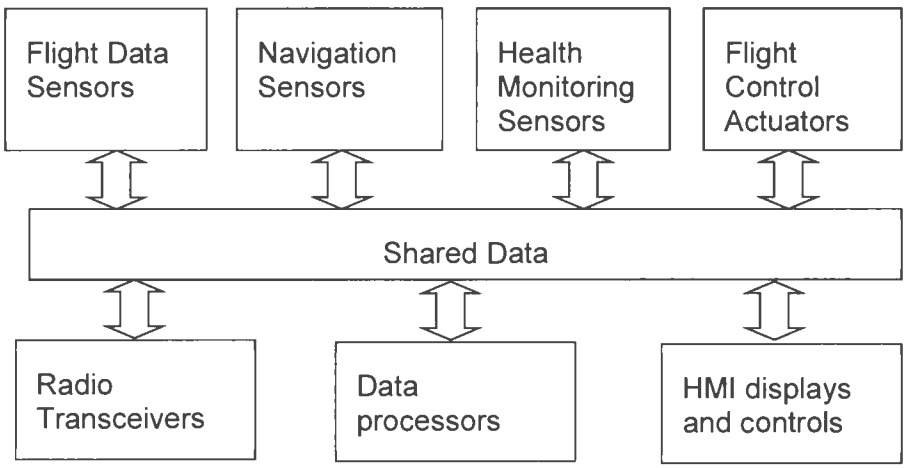

Figure 4: Integrated Avionics System

A comprehensive description of avionics architectures is described by Spitzer [7].

Objectives concerning the development and production of airborne electronic equipment typically relate to economy of scale. The service life of airborne electronic equipment is generally in the order of 15 years or more, which is much longer than the typical service life of consumer electronic devices. This means that the obsolescence of components and support systems, e.g. software development environments, can become a significant problem during the operational deployment of the system and places specific constraints on the selection of components during development stages. It must also be considered that airborne electronic systems are typically produced in limited quantities resulting in development costs forming a significant component of overall product cost.

Specific elements of an airborne electronic system are:

- Hardware;
- Software;
- Test and support equipment;
- Operating instructions;
- Training material.

Processes to develop airborne electronic systems must make provision for the creation and integration of all these elements. Due to the fact that development costs are

substantial, it is therefore imperative to streamline the methods by which development is accomplished.

3.2.2 Airborne electronic equipment development in South Africa

Long distances between South Africa and other industrialised countries, as well as significant distances between major centres in the country, stimulated the development of air transportation routes and associated services from the earliest days of commercial aviation in the country. However, largely due to economies of scale, South African industry has never embarked on the development and production of aircraft and airborne equipment on any significant scale.

Political conditions in the latter decades of the 20th century compelled the South African government of the time to encourage the development of extensive military engineering and manufacturing capabilities, which also lead to the establishment of expertise with respect to development and manufacture of aircraft and airborne equipment.

Political changes have negated most of this impetus, and the current situation is that a focus is placed on the retention of strategic capabilities, e.g. the development and manufacturing of electronic warfare and secure communications. In addition, a number of South African enterprises has the capability to perform modifications and upgrades to military aircraft and to integrate modern airborne electronic equipment into existing airframes.

In the light of the strategic view that this capability should be retained, it is important that the processes supporting this capability should be sustained and improved. This improvement aspect was an objective of this study.

3.3 Establishment of specific capabilities at Denel Aviation

The evolution of the processes for the development of electronic equipment for airborne use at Denel Aviation over the past twenty years is briefly described in this section. This account provides further background to the formulation of the research challenges, presented at the end of this chapter.

3.3.1 Development programs

Historically, Denel Aviation has been involved in many aspects of aircraft manufacture, including the production of airframes, engines, gearboxes and electronic systems.

Projects involving the development of airborne electronic systems containing embedded software and associated equipment are summarised below.

-Tactical coupler

Atlas Aircraft Corporation, the predecessor of Denel Aviation, started the development of a Tactical Coupler for an Automatic Flight Control System (AFCS) for the Advanced Development Model (ADM) prototype of the Rooivalk Attack Helicopter in 1986. This coupler worked in conjunction with a commercially available analogue autopilot (the SFIM AP-155) and provided an automatic hover mode, a sight mode that automatically steered the helicopter in the direction in which the main sight was aimed, and a control augmentation capability that enhanced the agility of the aircraft at low speeds. The system used a standard ruggedized computer developed for military use by a supplier in South Africa.

- Display processor

Denel Aviation also participated in the development of a Multi-Function Display Processor (MFDP) for a fighter aircraft, under the auspices of the main contractor[2] for the MFDP. This project started in 1988 and included systems engineering activities and software development.

- Mission planning systems

The company also started with the development of software for off-board mission planning systems in 1996. Note that although these systems do not meet with the definition for airborne electronic equipment, the development team used the same processes as were required for the tactical coupler and display processor, and contributed to the definition of the Denel Aviation processes.

- Digital AFCS for an attack helicopter

Development of a new fully digital AFCS for the Evaluation Development Model (EDM) prototype and production versions of the Rooivalk helicopter commenced in 1994. The design of the software algorithms was based on the control circuits in the AP-155 analogue autopilot, as well as on the algorithms previously developed for the tactical coupler described above. The system provided stability augmentation, attitude and heading hold, as well as a number of upper control functions. The digital AFCS utilised

[2] In some instances the text in this dissertation refers to information, e.g. names of customers, contractors and suppliers, where no further details can be provided due to reasons of confidentiality.

commercially available flight control computer hardware suited to the particular type of helicopter.

Figure 5: Rooivalk Attack Helicopter

- Navigation system for a transport helicopter

Development of a new navigation system for the Oryx medium transport helicopter was initiated in 2008. This navigation system formed part of an upgrade program to the helicopter fleet and was aimed at replacing obsolete equipment and provided improved navigation functionality. The system utilised legacy equipment, new off-the-shelf hardware and hardware developed specifically for the project. In this project, a significant amount of development work was shared between Denel Aviation and sub-contractors.

Figure 6: Oryx Medium Transport Helicopter

3.3.2 Development process evolution history

The situation concerning the development of airborne electronic equipment at Denel Aviation in 1986 had the following characteristics:

- Military aircraft were not subjected to airworthiness certification to the same extent as the current practise.
- Armscor invoked MIL-STD-490 [8] and MIL-STD-1521 [9] concepts in agreements, and payment milestones were arranged according to the typical baselines referenced by these standards.
- Systems engineering comprised the generation of a hierarchy of specifications, development and evaluation of a series of prototypes, executing acceptance tests against the specifications and a series of reviews and audits linked to development baselines, conducted according to prevailing military standards.
- Formal tests were witnessed by quality assurance representatives; no significant process or lifecycle data audits were performed.
- Documents were approved by passing the draft from one person to the next, suggested improvements were redlined by each reviewer and the author finally collated all comments and published the document. Configuration

management comprised the issuing of document numbers and management of a documentation repository.

- The company procedure manual was "paper based" and contained hardly any form of guidance with respect to the development of electronic systems for use on board aircraft and the associated lifecycle data.
- Reliance was placed on the capabilities of individuals rather than on applying a process approach to development. This paradigm existed throughout all levels of seniority in the company.
- Access to published information of any technical nature was limited to documentation procured by the enterprise; the use of the internet to disseminate information only became commonplace around 1995.
- Computer based systems to support the development process were not in widespread use.

The evolution of documented development processes at Denel Aviation, which are summarised below, must be considered against the abovementioned background. The process definition presented in the study is the culmination of the work undertaken in terms of process improvement, from 1994 up to the end of 2015.

The first project considered here is the development of the tactical coupler that was part of the development of the Exploratory Development Model (XDM) prototype of a new attack helicopter. Development activities at the hierarchical level of the helicopter program itself were controlled in accordance with a Systems Engineering Management Plan (SEMP), which described the development program and it detailed work methods such as rules for meetings and reviews. Software development was performed in a rapid prototyping environment where it was envisaged that a software certification process would follow at a later stage, and very little consideration was given to formal engineering process aspects.

At the time when the development of the digital AFCS commenced, it was understood that the software certification process had to be integrated into the development process, but only a few formal procedures relating to the development and qualification of electronic equipment containing embedded software existed in the organisation at that point in time. This restrained the ability of the team to plan the software qualification process efficiently. The need for a structured, documented process was identified, but no apposite resources were available to resolve the matter.

The development of mission planning software was performed under contracts to customers[3] external to Denel Aviation, and the organisation had to review its development processes to ensure compliance with quality assurance requirements from these customers. As a consequence of this review, the author was tasked to initiate a process improvement project.

A technology transfer program between Saab AB of Sweden and Denel Aviation provided significant assistance in terms of this process improvement exercise as part of offset agreements in terms of the procurement of the Swedish JAS39 Gripen front line fighter aircraft by the SAAF. One of the desired outcomes of this program was the establishment of a local capability in South Africa that could modify software on board the South African versions of the JAS39 Gripen fighter aircraft, in compliance with Saab's quality and airworthiness standards. The outcome of this participation was a documented process for the development of airborne electronic systems, which was issued in 2004. For reference purposes in this study, this process is referred to as the Airborne Electronic Systems Development Process (AESDP). This process was developed under the auspices of the author and under guidance from specialists employed by Saab.

The AESDP presented a workflow structure together with procedures for the execution of detailed tasks. The AESDP contained elements of MIL-STD-498 [10] as well as elements that satisfied RTCA/DO-178B [11] requirements. Principles described in ANSI/EIA-632 [12] were also incorporated in the AESDP. The AESDP had to be compatible with change control and configuration management tools used by Saab AB at the time. As no order for work on the Gripen aircraft was placed subsequent to the acceptance of the AESDP by Saab, the system was not integrated with the quality management system (QMS) at Denel Aviation, and consequently no further support was allocated to the improvement and maintenance of the AESDP. Elements of the AESDP were applied to the development of the AFCS, where applicable.

The AESDP was a major input to the MEng study [13] that preceded the work described in this dissertation. The process described in the study was used to develop the statement of work for the development of the upgrade to the Oryx navigation system, described in section 3.4.3.

During the development of the navigation system for the medium transport helicopter, procedures relating to the process definition that existed at the time were applied and

---

[3] See footnote 2.

improved, and new procedures and work instructions were developed where shortcomings were identified.

After the completion of the development of the Oryx navigation system, the process definition was further improved, resulting in the process definition that is the subject of this study.

A timeline for the development of the process definition is presented in Figure 7.

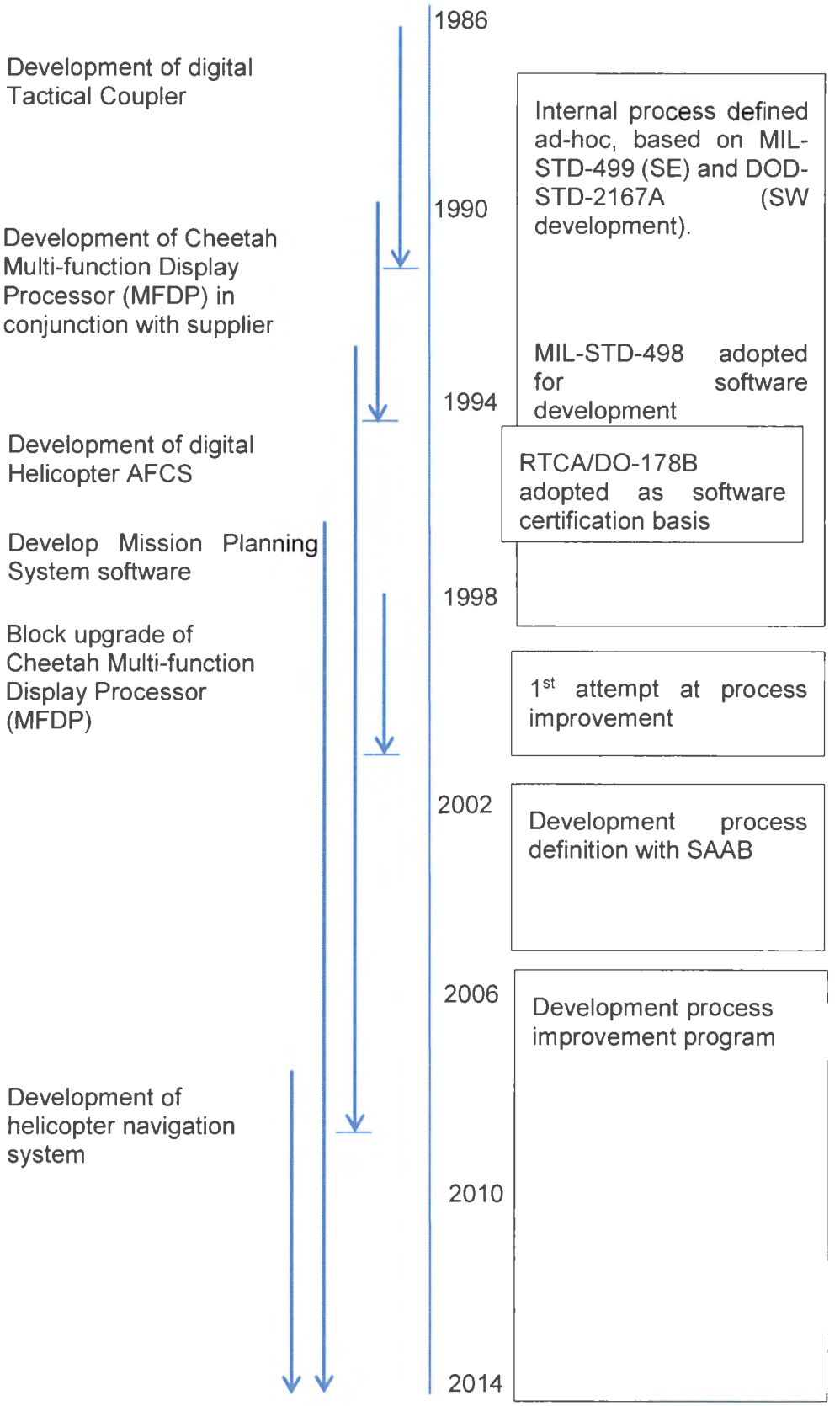

Figure 7: Development history timeline

3.3.3 Assessment

As explained above, the need for formalised guidance in terms of the development of airworthy systems employing embedded software was identified during the development the digital AFCS. No comprehensive process description, which aimed at meeting military qualification standards, airworthiness requirements, software certification requirements, and software quality standards, which could readily be implemented, could be identified from accessible sources at the time.

An objective of the AESDP, described in the preceding section, was to provide a frame of reference for all the activities that were to be performed, as well as to provide the details for the execution of all the tasks related to the development of airborne electronic equipment. This included detailed descriptions of all documentation required. The AESDP had to take into account contractual as well as airworthiness considerations, and had to be aligned with best practises as published in relevant industry standards and guidelines. In order to incorporate all these aspects, the development of the AESDP required a considerable effort in terms of labour that involved a number of knowledgeable people. This fact in itself also indicates the difficulty of the task of determining the details for an engineering process with respect to a specific application. The motivation for the MEng study [13], which was initiated during the final stages of the development of the AESDP, was to establish a documented record of the rationale behind the design of the AESDP. The MEng study re-confirmed the fact that the development process for airborne electronic equipment, specifically when it involves developmental flight-testing, is multi-facetted and extensive.

The observations expressed below were made in the course of the formalisation of the development processes throughout the stages described in the preceding section.

It must be noted that the effort to produce a system or product, such as those considered in this study, includes the management of resources (such as time and money), the application of skills and knowledge involving various disciplines, the utilisation of special tools and equipment, assurance of the adherence to statutory rules and regulations, and the management of interactions between acquirers and suppliers. It soon became evident that the quest to describe all these activities in a coherent description is impractical. Those aspects related specifically to the definition and qualification of the product, i.e. the "core" engineering process, needed to be segregated from the other project aspects in order to identify a finite extent of the definition of the process to be described. This task was problematic, as different

stakeholders did not agree on all aspects of the segregation. The lack of a clear identification of the scope of the development process, grounded in sound engineering principles, hampered efforts to formalise the engineering processes.

A problem associated with this obstacle of unclear process scope, was that references that were consulted at the time of developing the AESDP described different engineering processes at e.g. the level of development of software, development of electronic hardware and at the level of the integrated system. This resulted in a complex process description, which required simplification.

It was further found that requirements for tasks to be included in the development process were stated in a number of inconsistent sources, e.g. systems engineering standards and airworthiness recommended practises. The unequivocal identification of the activities that need to be incorporated in the process design, considering the requirements from various sources, proved to be difficult.

Furthermore, these activities are not executed in isolation, but a significant amount of interaction occurs between them. These interactions include the hand-over of data and other artefacts between activities or different areas of responsibility and needs to be managed astutely to prevent confusion and disorder. No model, which met with requirements of all the stakeholders that described these interactions unambiguously, could be identified in the process formalisation action.

It is also important to note that the outputs of the development processes must be of such a standard that it will stand up to scrutiny in a court of law, particularly due to the fact that failures of airborne systems can lead to incidents with legal consequences. Therefore, the correctness and integrity of the data items produced during the development process are of prime importance.

The difficulties identified during the development of formalised engineering processes at Denel Aviation are summarised below.

> *The scope of the core engineering process, applicable to the development of each logical system element, was not distinctly defined for airborne electronic equipment and lead to complications in developing a process definition.*
>
> *No comprehensive and consolidated definition of the activities for the development of airborne electronic equipment in the South African military context, stated in terms of inputs, actions and outputs, existed. This complicated the development process formalisation task.*

No clear-cut model to describe interactions between engineering process activities, suitable for the development of airborne electronic equipment could readily be sourced.

The control mechanisms to manage the traceability between data items and to ensure the correctness and integrity of the data produced during the development process proved to be insufficient in terms of meeting with airworthiness requirements.

3.4 Retrospective assessment of development programs

Additional motives for the detailed formulation of the development process that formed the basis for this research originated from concerns encountered during the execution of the development programs introduced in section 3.3.1. A retrospective assessment of two major programs executed by Denel Aviation is presented in the following sections, with the objective of substantiating the research challenges that form the basis of this study.

3.4.1 Introduction

The two projects described below each exceeded at least five years in development time, and included the development of hardware, embedded software, and enabling products. The development activities included the specification and procurement of off-the-shelf equipment and management of sub-contractors.

An important characteristic of these programs, of particular importance in terms of the assessment of the development practises, is that in both instances the system-of-interest was an upgrade of a previously existing system. The objectives of the programs were to implement improved technologies and to add functionality that enhanced the operational capabilities of the platforms for which they were developed. This implied that the intended purpose of the systems-of-interest was well understood, and that the functional and performance requirements for these systems were largely known.

3.4.2 Development of a digital AFCS for an attack helicopter

3.4.2.1 Introduction

The development history of the digital AFCS provides an insight into the highly technical nature of typical airborne electronic systems as well as into the complexity of

the total task to declare a system intended for airborne use airworthy. It also provides insight into the complexities of the development infrastructure.

3.4.2.2 System overview

The AFCS on board the Rooivalk helicopter provides stability augmentation of the helicopter through rate feedback in the roll, pitch and yaw axes, thereby improving its handling qualities. It also provides pitch and roll attitude hold, heading hold, turn co-ordination and collective cross coupling reduction control modes to enable the crew to maintain selected attitudes and heading, and change speed, heading or altitude with a minimum of effort while their attention can be focussed on other tasks. It also provides control augmentation functionality, by means of feed-forward loops and control feedback loop gain scheduling in the cyclic channels, to enhance the low speed control response of the helicopter during tactical manoeuvring while performing nap-of-the-earth (low flying) operations. In addition, the AFCS provides functionality by which the flight path of the helicopter can be controlled without any control input from a pilot, where it steers the helicopter according to a pre-defined flight plan (navigation mode), maintains barometric altitude (altitude hold mode), aligns the helicopter's centreline with the line of sight of the main sight system (sight mode), and maintains a "zero groundspeed" hover (hover and height hold modes). These so-called "upper mode" or "coupled" flight path control modes, as well as the control augmentation modes, are purposely selected by the crew, whereas other modes are active whenever the AFCS is engaged.

The AFCS forms part of an integrated avionics system as described in section 3.2.1 and receives attitude, rate and acceleration data from sensors which form part of the on-board navigation system, mode selections and steering commands from the helicopter's mission management system, and flight control position data (cyclic and collective levers and yaw pedals) from the manual flight control system. It applies flight control commands to the main rotor swashplate and the tail rotor spider by interfacing to actuators in the manual flight control system. The AFCS also reports system status information to the mission management system and the health monitoring system, for annunciation to the pilot and for the recording of failure data for use by the maintenance crew. A schematic diagram of the AFCS is presented in Figure 8.

The flight control computer hardware was procured off-the-shelf (from SAGEM, France) and a dedicated mode controller was developed in-house in participation with local subcontractors. All on-board embedded software for the AFCS was developed in-house.

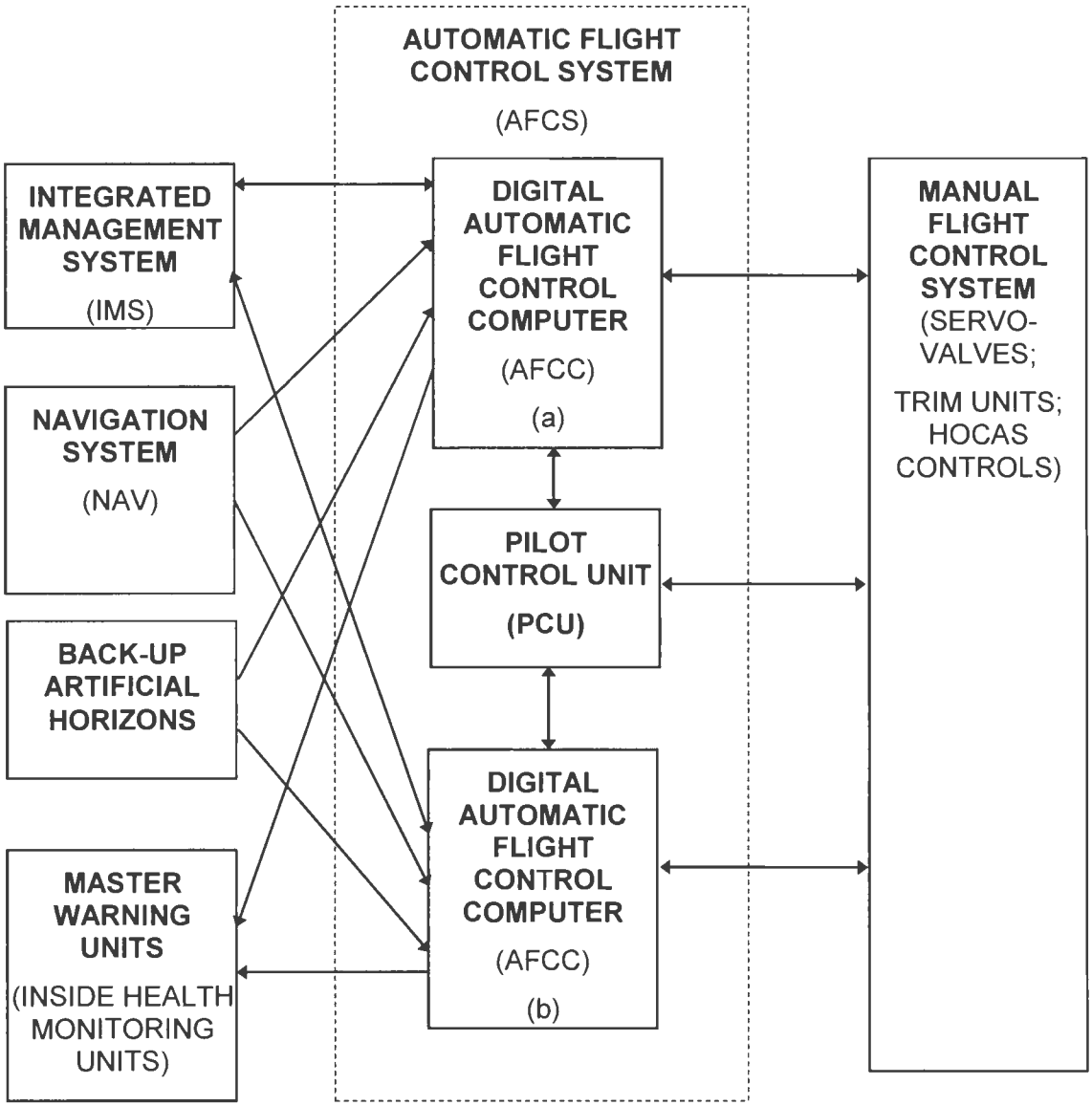

Figure 8: AFCS System Architecture

3.4.2.3 Development process

Although the development of the AFCS involved many facets of engineering processes, notable aspects of the development process that merit further consideration were the systems engineering approach, the integration and testing processes, and the process control mechanisms.

- Systems engineering approach

At the stage of the Rooivalk development program when the development of the EDM prototype was contemplated, program stakeholders decided that the functionality, performance and airworthiness of the helicopter and its subsystems would be

demonstrated against DEF STAN 00-0970: Design and Airworthiness Requirements for Service Aircraft VOL 2- Rotorcraft [14], a UK Ministry of Defence standard for the design and airworthiness of helicopters, and to adopt RTCA/DO-178B: Software Considerations in Airborne Systems and Equipment Certification [11], as the basis for certification of software.

However, during initial stages of the development of the EDM prototype, which was the first Rooivalk prototype model to employ a new digital AFCS, the project focus was primarily decided by marketing objectives, therefore the delivery of a demonstrable prototype took priority over qualification and certification considerations. It was tacitly understood that qualification and certification processes would follow in due course, and that a "prototyping" approach would be followed in the interim without compromising the integrity and safety of the aircraft.

This prototyping approach also applied to the methods for developing the digital AFCS, which commenced in June 1994. This methodology provided a functional "prototype standard" AFCS, which was available for flight-testing of the EDM prototype at the time when the construction of this aircraft was completed in November 1996. However, this system did not meet the airworthiness requirements for an operationally deployable aircraft.

After initial demonstration of the capabilities of the Rooivalk EDM prototype, qualification and certification of the helicopter became the main focus of the development program, and eventually the AFCS was granted full airworthiness approval in April 2011. The aircraft has since been deployed successfully in operational tasks by the South African Air Force.

The development of AFCS software code was mainly based on pre-cursor information, i.e. the "top-down" route (working from system specification to architectural design to detailed design to software coding) was not followed rigidly. The pre-cursor data, as well as prior knowledge of similar helicopter flight control systems, were applied to develop source code from the outset, and this source code became the de-facto design reference for practical purposes, during early development stages of the project.

An effort was undertaken to develop the required lifecycle data as required by RTCA/DO-178B [11], for qualification and airworthiness certification purposes. However, no templates of the applicable documents that could readily be used were available. The guidelines in RTCA/DO-178B [11] are by and large generalised, leading to individual interpretations and consequent inconsistencies of document attributes. The authors of the lifecycle data items had to develop formats of documents to the best

of their knowledge due to the lack of documentation instructions. Where it was deemed practical, MIL-STD-498 [10] was used for guidance.

The final set of lifecycle data for the project was produced by means of considerable reverse engineering, using the software source code and system interface data as references. This exercise of documenting and reviewing lifecycle data required an extensive investment in terms of labour and time. The magnitude of the task can be appreciated if it is considered that the source code exceeded 100,000 lines of code.

- Integration and testing

Initial integration testing of the AFCS software was performed in a laboratory setup that employed static simulations of AFCS inputs, e.g. the generation of aircraft rates and accelerations in an appropriate electronic data format such as ARINC 429. This setup is shown in Figure 9. Further system integration testing was performed on an Avionics Integration Bench (AIB) where the AFCS interfaced with actual avionics hardware to demonstrate the correct operation of the data interfaces. Furthermore, a standard SA 330 J Puma helicopter (see Figure 10) was modified to provide for the installation of the major elements of the AFCS. Flight-testing of the AFCS on board this test aircraft commenced in December 1995. Most of the AFCS control algorithm design evaluation was performed on this aircraft. At an appropriate stage, the digital AFCS was implemented in the Rooivalk EDM prototype aircraft to enable flight-testing of all aspects of the helicopter[4] as well as for further testing of the AFCS, specifically related to the dynamic characteristics of the different airframe. It was found that the different airframe and slightly larger weight of the Rooivalk did not require major adjustments of AFCS control parameters, and no significant amount of flight test time was expended on resolving AFCS problems on the Rooivalk EDM prototype.

[4] The AFCS is an integral part of the flight control system, i.e. it needs to function correctly to render the aircraft controllable throughout its envelope.

Figure 9: AFCS Test bench (static stimulation)

In the meantime, further developmental testing of AFCS upper mode functionality, such as automatic hover and height control, continued on the Puma test aircraft for a significant period of time. As the Puma was not fitted with a main sight system, the AFCS sight mode operation had to be resolved on the EDM prototype.

Figure 10: SA330 Puma test aircraft

The availability of appropriate hardware and the participation by Denel Aviation in a project to develop a local (South African) modelling and simulation capability aimed at aviation technology provided an opportunity to develop a pilot-and-hardware-in-the-loop helicopter simulator, as shown in Figure 11. The potential of this system to support the development of the AFCS was recognised and the system was extended to accommodate integration and verification testing of the AFCS. This pilot-and-hardware-in-the-loop simulator based test system was also used to investigate AFCS issues that were reported during flights on the Rooivalk during the initial stages of the deployment of the aircraft into service, with reasonable success.

A high-fidelity software model of the dynamic behaviour of the helicopter was employed in the hardware-in-the-loop system, but due to technical constraints, this simulation could not execute in real time when even moderate rates of change of dynamic data had to be handled. This reduced the scope of application of the test system at the time.

Figure 11: Helicopter hardware-and-pilot-in-the-loop simulation system

Dedicated AFCS related verification testing on the EDM prototype was largely devoted to the demonstration of pilot intervention delay times with respect to servo hard-overs and trim runaway fault conditions, as well as demonstrating that these fault conditions did not pose a hazard with respect to structural overload of aircraft structural components.

The pilot-and-hardware-in-the-loop simulator based test system, as described above, was used in particular to perform safety related testing that could not be performed in flight, e.g. the response of the system upon simulated sensor failures. A notable issue,

which arose when this system was implemented, was that the simulator system also had to be "qualified" in order to validate test results produced by the system. The same problem of the reverse-engineering of data from the details of a system which was developed along a prototyping approach due to an insufficiently defined development framework, as described above in terms of development of the AFCS, manifested in this instance.

- Development process controls

On the AFCS project, the formal configuration management process was only executed on the air vehicle level. This meant that the configuration control of airborne software was managed through the application of the air vehicle Engineering Change Proposal (ECP) process, i.e. there was not a formal configuration management process implemented to manage the development of elements of the system at the AFCS layer of the system hierarchy. Control of development process outputs at the AFCS layer was limited to the following:

Documents were verified for correctness through the informal handing of a document from one person to another for redlining, collation of comments and the eventual editing of the final document by the author, after which the author carried a copy of the document around to designated personnel for signatures.

The process for the approval of documentation evolved from the "redlining" process to the "Fagan inspection" process (introduced via the AESDP), which markedly improved the efficiency by which documents were produced and released. However, the improved process still fell short of complete validation of contents of a document, as the focus was more on superficial quality aspects of the document, such as formats, spelling and readability. For example, no checklists were available to assess the technical contents of documents.

Formal quality assurance audits were conducted on the AFCS development project, although only a very limited number of internal standards, policies and procedures were identified and available against which process adherence could be measured, or against which the outputs of particular activities could be evaluated.

MIL-STD-1521 [9] type design reviews were also only performed on the air vehicle layer, and there was very limited interaction between the AFCS design team and the contracting agency, as compared with the significant interactions between the agency and the systems engineers during the development of the Oryx navigation system, as described in section 3.4.3 below.

3.4.2.4 Assessment of the AFCS development process

Notable aspects from the development of the AFCS to be considered in the research in this study are the development strategy, the software development process, the qualification and airworthiness certification process, and the change management methodology.

- Development strategy

The strategy and the process of developing lifecycle data, as explained above, enabled the AFCS functionality to develop rapidly, but it had detrimental consequences in terms of broader program objectives. In this "prototyping approach" strategy, most of the source code for the AFCS was produced without formal design or requirements documentation, as sufficient information was available from other "informal" sources, and the engineering tasks were executed by experienced and knowledgeable personnel.

An adverse consequence of this methodology was that the source code typically contained functionality that was not reflected in other design documentation. If documents were not specified as project deliverables, e.g. a specific system specification, there was no urgency in generating them, leading to a consequent lack of general visibility on system design aspects, including a lack of design meta-data. It is also significant to note that, also due to this prototyping view, insufficient attention was given to the preparation of and adherence to detailed development plans during the initial stages of the program.

The lack of data that describe specific system attributes meant that integration tasks could only be performed effectively by persons conversant with the source code and who had intimate knowledge of the analogue autopilot, which was replaced by the new digital AFCS. The general lack of accessible design data, e.g. the specific implementation of flight control functionality, compelled other members of the development team, e.g. flight test crews, to continuously consult with the persons who developed the source code in order to plan their activities, a difficulty which could have been avoided if the design information was more accessible. Also, note that this "localised knowledge pool" in itself posed a significant program risk that could be introduced by the unavailability of this limited number of knowledgeable people due to unforeseen circumstances.

Another significant consequence of inadequate lifecycle data was that it hampered the eventual processes to obtain airworthiness approval for the AFCS. This delay restricted the operational use of the AFCS, as the assumption had to be made that the

"uncertified" software could cause anomalous behaviour of the system. This risk could only be mitigated by requiring the pilots to use all upper modes, e.g. automatic hover, "attentive, hands-off". This meant that they could not rely on the AFCS to control the aircraft while they engaged in other tasks. The problem of insufficient structured documentation of requirements and design data remained a large program risk until the final qualification and delivery of the system, when the requirements and design data eventually fully reflected the functionality implemented in the source code.

It is therefore evident that the "prototyping" approach was effective in terms of realising a functional system early on in the program, but that it introduced risk and was detrimental in terms of the efficiency by which the final qualification of the system was achieved.

> *A root cause of this inappropriate strategy was that at the outset of the project there was no formally documented scheme describing the necessary activities essential to the accomplishment of a certified system.*
>
> *A further cause of the flawed strategy was that there was no suitable control mechanism to ensure correct execution of defined engineering activities, even if these definitions existed.*

- Software development

The AFCS project was the first effort undertaken by Denel Aviation to develop embedded software intended for certification against RTCA/DO-178B [11], and important lessons were learned in the process. Therefore, the software development aspect is considered as a separate topic in this assessment.

A very important aspect to consider is that the methodology for the development of systems containing embedded software differs from the conventional method of developing a product, which was normally accomplished through the use of a series of prototypes. In the case of the development of the AFCS, the embedded software was improved and debugged throughout the development cycle until the final version received airworthiness approval. This method contrasts with the typical development involving e.g. a printed circuit board in the era before computer aided design became the norm. In this example of a printed circuit board, the circuit was laid out on a prototype board, and errors and deficiencies were corrected by on-board modifications until the desired functionality was achieved. A production version of the board was then developed, incorporating all changes and modifications, and the prototype board was discarded. However, when software is updated, software design and coding faults are typically removed by following a software change control process. This process

continues until no known errors are present in the code, and this process is normally associated with copious testing which instils sufficient confidence in the final code base to declare it fit for use. Therefore, the notion of discarding prototype code and generating correct code is injudicious. From the development history of the AFCS it is also clear that the final outcome of the development process was a "first article" and that no prototypes of the digital AFCS were developed, in the traditional sense. This first article fully represented the design of production versions of the AFCS.

The notion of a lifecycle model where a first article is developed from the outset is different from the sequential methodology promoted by e.g. RSA-MIL-STD-3, which relies on a number of prototypes during the development cycle, to achieve different objectives. This model is entrenched in the military industry, and is applied by Armscor in their management of acquisition activities.

As most projects undertaken by Denel Aviation are in accordance with agreements with Armscor, Denel Aviation has to comply with Armscor's acquisition processes. At the same time, as shown above, a development paradigm based on the engineering of a first article item without the use of formal prototypes is more apt to the development of airborne electronic systems.

> *It can therefore be concluded that the development process was impeded by the conundrum of a "first article" engineering approach vs. an engineering approach that uses the conventional series of prototypes for development.*

- Qualification and airworthiness certification

The problem of attempting to perform the qualification process "after-the-fact" was highlighted in section 3.4.2.3. In order to quantify this problem, consider that the qualification process essentially consists of the following elements:

1. Identify and validate the complete set of requirements and standards against which the system must be developed and against which compliance must be shown.
2. Determine the methods by which compliance with the requirements and standards are to be demonstrated.
3. Develop the system, as well as the test cases and procedures that will be used to demonstrate the specified compliance.
4. Perform the compliance demonstration tasks on a representative specimen of the system-of-interest when a controlled baseline for the system is established,

to prove that the system as well as the lifecycle data describing the system's attributes comply with the identified requirements and standards.

5. Resolve non-compliances and discrepancies.

The success of the qualification and certification process is dependent on the correctness, completeness, and integrity of the requirements and standards against which compliance is demonstrated, and the suitability of the arrangement that is used to perform the verification tests. In addition, emphasis is placed on demonstrating traceability between requirements, design and implementation details, as well as on the demonstration of verification coverage of design aspects and requirements. In the case of the development of the AFCS, these verification and validation processes could not be implemented correctly as long as a complete set of correctly formulated system and software requirements for the AFCS was not available. Adequate test coverage could not be proven against approved specifications to substantiate claims of safety of operation of the system as long as these specifications were not mature.

Another important observation regarding the qualification process is the fact that it took a significant effort to reverse engineer the qualification lifecycle data from the source code was indicated in section 3.4.2.3. The situation of processing large amounts of data in order to produce qualification evidence could possibly have been ameliorated if smaller blocks of functionality (implemented through software), e.g. the primary control functions such as attitude and heading hold, were qualified and then released for use, while development and qualification of the upper modes, e.g. automatic hover, were completed at a later stage. This implies a lifecycle model and associated baseline control system that differs conceptually from the entrenched "waterfall" model where verification is performed after full integration of all the system elements has been completed. In the type of process where the outcome is in the form of a first article, the development process should allow for modifications to be applied with limited restrictions until a stage has been reached where a representative item, at a controlled configuration baseline, can be subjected to formal verification tests. This aspect is described in more detail in the assessment of the development process for the Oryx navigation system.

Therefore, in summation, it can be stated that the difficulties in qualifying the system were firstly caused by a lack of definition of processes for the generation of the lifecycle data required to meet with the stipulations of the acquisition and airworthiness processes, at the outset of the project.

The second root cause of the observed qualification problems was the utilisation of an inappropriate lifecycle model for the development of a system employing embedded software.

- Change management

The use of the air vehicle layer ECP based configuration management process was found to be cumbersome for implementing small code updates required to fix problems during integration testing. On the other hand, as this process was mainly structured to manage changes to mechanical components and systems on the aircraft, it did not provide effective control over the development of lifecycle data associated with the development of embedded software. In particular, the configuration management of the source code of the AFCS was, for the largest part of the project, under the direct control of the developers of the code. This lead to inefficiencies in terms of utilisation of resources, as well as difficulties in ascertaining the integrity of the software.

In short, the management of the lifecycle data during the development of the AFCS was ineffective and caused inefficiencies as well as certification difficulties.

3.4.3 Development of a navigation system for a medium transport helicopter

3.4.3.1 Introduction

The project to modernise the avionics system on board the Oryx fleet utilised experience from producing the AESDP, as well as from the development of the AFCS, described previously. However, a number of shortcomings in terms of the efficiencies by which the development process was executed were identified. In the sub-sections that follow, the development process of the upgraded navigation system for the Oryx is described briefly, followed by assessments of relevant aspects that lead to the identification of development process shortcomings.

3.4.3.2 System overview

The Oryx avionics modernisation program included the installation of a new communication suite and the replacement of obsolete navigation equipment. The upgraded elements of the navigation system replaced an obsolete navigation computer, GPS receiver and antenna. It also provided an HMI functionality in support of the communications system to allow for the editing of communications channels and text communication messages. A high-level architecture of the upgraded navigation system is shown in Figure 12. The Mode Control and Display Units (MCDUs) perform

navigation and guidance computations and display navigation and communications data to the crew. The Data Transfer Unit (DTU) is used to transfer data from a mission planning system to the navigation system via a Personal Computer Memory Card International Association (PCMCIA) module, a type of memory card. The Avionics Interface Unit (AIU) interfaces the modern MCDUs with the legacy equipment on the helicopter (showed in grey in Figure 12) and performs data conversion with respect to new equipment where required. The AIU was specifically developed by Denel Aviation for this project. The MCDUs are produced by CMC Esterline in Canada. CMC also provided a significant portion of the on-board software, while Denel Aviation was responsible for the development of the HMI and navigation software that pertained specifically to the Oryx avionics architecture and user doctrines.

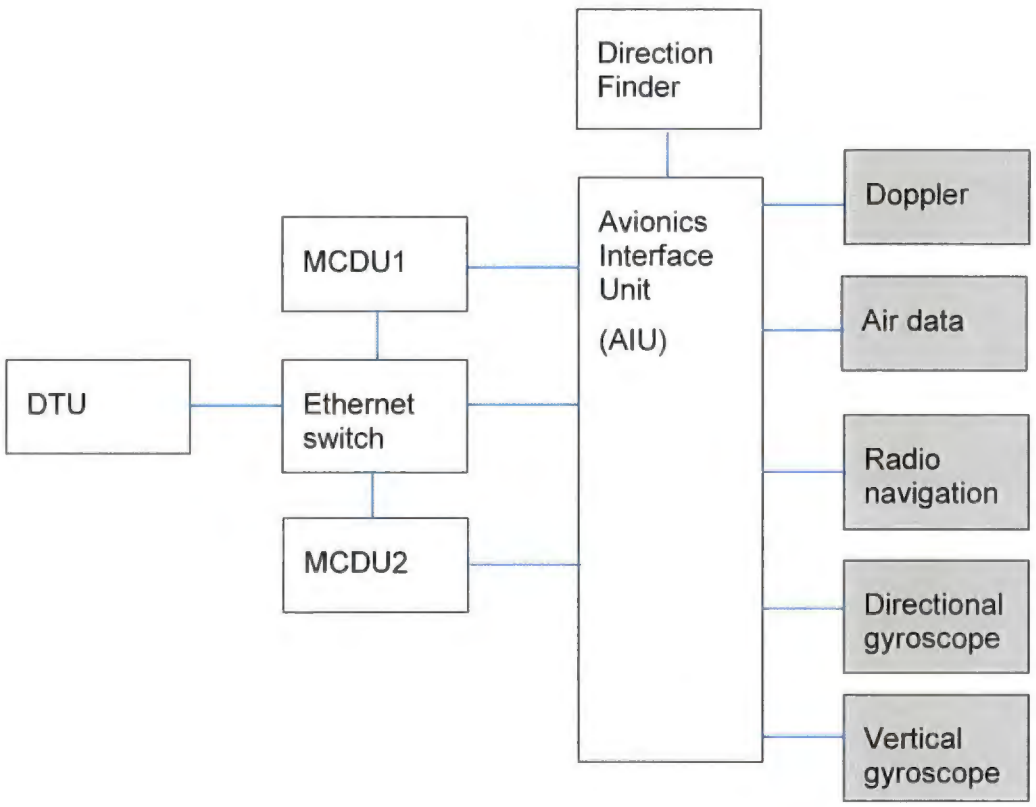

Figure 12: Oryx navigation system

3.4.3.3 Development process

The Statement of Work (SoW) which formed a basis for the agreement between Armscor and Denel Aviation for the upgrade of the Oryx fleet was based on an outcome of the MEng study [13] that preceded this research. The project was executed in accordance with this SoW to a large extent. This SoW was comprehensive in terms of

the specification of specific activities and their required outcomes, but did not specify detailed methodologies and techniques.

The contracting agency (Armscor) invoked RSA-MIL-STD-3 [15] as the reference for the systems engineering effort on the project. This standard prescribes the use of a series of prototypes and associated baselines, and MIL-STD-1521 [9] style reviews and audits to accomplish the development of a system. A Systems Engineering Management Plan (SEMP) and Test and Evaluation Master Plan (TEMP) were prepared to integrate the SoW with the RSA-MIL-STD-3 defined milestones and baselines.

A requirements-based process was used for the development of the navigation system. An in-house requirements management tool (based on Microsoft Excel®) was developed during the execution of the project. This tool was also used to capture the results of a Functional Hazard Assessment (FHA), and to list proposed means of compliance (e.g. analysis, rig test, ground test or flight test). In due course, the tool was used for the management of traceability between requirements, implementation details and verification data.

The simulator-based test system, described in section 3.4.2.3, was adapted for this program and was used for the laboratory integration of the upgraded avionics system.

An Oryx helicopter was upgraded with the new system and fitted with flight test instrumentation. This aircraft served as a prototype for the development and verification flight tests, which amounted to approximately 200 flying hours. A supplemental type certificate for the Oryx, incorporating the upgraded avionics, was issued in March 2014.

3.4.3.4 Assessment

- Development process model

As in the case of the AFCS, this navigation system program also produced a first article without a formal prototyping stage, and the methodology was supported by the SoW mentioned in section 3.4.3.3. However, the Armscor processes were not aligned with this methodology, and this lead to discrepancies in terms of the expectations of design reviews and configuration audits, as prescribed in the classical process, on which the Armscor process was based.

> *The lack of a process description that is commensurate with the development of a first article without the use of prototypes as well as with the processes*

applied by Armscor lead to difficulties in executing the programme.

- HMI definition

A significant component of the Oryx navigation system modernisation programme revolved around the presentation of navigation and flight guidance data on display panels in the cockpit. In order to ensure that the requirements of the end users, i.e. pilots and flight engineers, were addressed in the HMI design, workgroups were arranged where aircrew participated in the definition of the display functionality. The result was that most of the outcomes of the workgroups were already in the form of display design features, rather than in producing requirements, as the concept of requirements formulation was alien to most of the key participants. This design was captured in a HMI design document. A reverse engineering exercise was conducted to determine actual HMI requirements using the HMI design document. The objective of this exercise was to determine what functionality had to be implemented, whereas the HMI description already defined how the functionality should be implemented. These requirements formulations were required to support the system safety process. A prudent methodology should have focussed on establishing the HMI requirements before embarking on a detailed design exercise.

> *This further underlines the problem of ineffective execution of development tasks due to an insufficient definition of development process activities and expected outcomes, formulated in terms that are understood by all stakeholders.*

- Integration and test

A distinction was made between integration tests and verification tests on the project.

Integration tests were executed with the objective of resolving integration problems and to detect and correct design and implementation deficiencies. This implied a fluid baseline, i.e. the baseline changed continuously due to regular updates to requirements, design, source code and hardware to implement corrections.

Verification test results, in general, are used as evidence to demonstrate that a set of requirements are satisfied by a synthesised system with specific attributes. Verification tests are normally formalised and witnessed by quality assurance representatives from a contractor and a contracting agency.

In principle, verification tests can only be performed on controlled baselines. If the design of the system is altered in order to correct a deficiency, some system attributes may change, which could imply that the behaviour of the system may have been

affected and that previous test results on the system may be rendered invalid. Therefore, it is prudent to ensure that the system contains no defects at the stage when it is subjected to verification tests. The practical approach is to subject the system to dry runs of the verification tests before the formal test is performed. These dry run tests may detect latent defects that were not detected during integration testing and may also identify errors in test procedures, which can be corrected prior to formal testing and the establishment of the associated baselines. Due to the high costs of flight tests, it is normally endeavoured to keep the amount of flight testing to a minimum, and it is impractical to execute dry runs of all planned verification flight tests. Two problems associated with this tenet were identified during the flight test campaign on the project.

Firstly, some parameters can only be determined by means of flight tests, hence tests were conducted on certain systems during the integration flight test stage to confirm correct operation and to identify problems, if any were identified. Testing the VHF radio installation on the aircraft provided an example of this problem. In the case of investigating whether the installed performance of the VHF radio antennas conformed to the relevant specifications, it was found that it was indeed the case and no modifications were required with respect to the antenna installations in order to improve performance. These tests were, however, conducted prior to the establishment of a formal test baseline, and reasoning had to be put forward as to why the tests did not have to be repeated during the verification test phase. In this example, two arguments settled the dispute: an analysis has shown that no consequent updates to the aircraft and systems could have an effect on the installed performance, and the aircraft was commanded by an Air Force test pilot on the particular flight tests, who represented the user *de facto* in a quality assurance capacity. This problem could have been avoided if a baseline control process was in place on each significant layer of the aircraft system hierarchy, rather than on only the aircraft layer, as was the case.

The second difficulty, in terms of verification testing, arose when software implementation problems were only revealed during the verification flight test phase. It had to be demonstrated objectively that updates to the software would not affect test results that were recorded during verification flight tests up to that stage. This problem highlighted the need for understanding and managing information feedback in the development cycle through prudent baseline management associated with an appropriate lifecycle model.

> *It can thus be concluded that a lack of a suitable baseline control mechanism, associated with a suitable lifecycle model, had detrimental effects on the efficiency by which the programme was executed.*

3.5 Identification of research challenges

To summarise: A process for the development of platform specific airborne electronic systems, within the environment where the development projects are to be performed, is developed by applying the design cycle of the DSR process, as will be described in Chapter 5 of this dissertation. The establishment of a theoretical foundation for the design of the process will be established by the examination of the applicable body of knowledge, by applying the DSR rigour cycle. In this instance, this entails the literature study, which is presented in Chapter 4. The requirements that form the basis for this literature study are formulated below. The requirements are based on assessments of relevant aspects of the environment for which the process is intended, that were described in this chapter, as part of the DSR relevance cycle.

In the formulation of the research challenges below, the problem to be addressed is summarised first, followed by a corroboration of the identified problem by referring to the project histories and retrospective assessments presented in this chapter. The specific aspects to be addressed by the literature research are then specified in the formulated research challenge statements.

3.5.1 Scope of engineering activities

The fact that the exact boundaries of the engineering processes, in terms of tasks and desired outcomes on the different layers of the system hierarchy, were not clearly understood by all stakeholders of the projects described in the retrospective assessments, was highlighted as a problem which affected the effectiveness by which projects were executed.

In section 3.3.3, it was pointed out that a problem encountered in formalising the engineering processes within Denel Aviation was that the scope of the core engineering process, applicable to the development of airborne electronic equipment, was not clearly defined. This included an observation that the models which applied to the breakdown of a system in its constituent elements are not rationalised sufficiently.

It follows that, when setting out to design a process, the boundaries of the process, as well as the interfaces with other relevant processes, must be established. Therefore, the research challenge associated with the problem of an inadequately defined scope of the engineering activities is formulated as follows:

> *The scope of the engineering process for the development of airborne electronic equipment is not pertinently established.*

3.5.2 Definition of activities and tasks

The retrospective assessments have shown that qualification and airworthiness requirements, conditions imposed by the acquirer, test and integration constraints (including development flight-testing), as well as recommended practises adopted by the aerospace industry, were not all clearly identified at the time when the projects were planned. This lead to difficulties and inefficiencies in the eventual execution of the projects.

The fact that a lack of a comprehensive and consolidated definition of the activities for the development of airborne electronic equipment in the South African military context(stated in terms of inputs, actions and outputs) existed complicated the task of the development formalisation task, was described in section 3.3.3.

In section 3.4.2.4 it was stated that a root cause of the inappropriate development strategy, followed in the development of an AFCS for the Rooivalk helicopter, was that there was no formally documented scheme describing the necessary activities essential to the accomplishment of a certified system at the outset of the project.

It was also pointed out in section 3.4.2.4 that difficulties in qualifying the AFCS were caused by a lack of definition of processes for the generation of the lifecycle data required to meet with the stipulations of the acquisition and airworthiness processes.

Section 3.4.3 described the problem of the ineffective execution of development tasks on the Oryx navigation system upgrade programme due to an insufficient definition of development process activities and expected outcomes, formulated in terms that are understood by all stakeholders.

It must therefore be ensured that the activities and tasks imposed by all relevant areas of interest are identified. This leads to the next research challenge:

> *The activities and tasks required for the development of platform specific airborne electronic systems are not identified explicitly.*

3.5.3 Development process framework

The lack of a comprehensive framework recognised by all stakeholders, describing development process activities and the associations and interactions between these activities, was shown to hamper the planning and the control of the development processes.

In section 3.3.3 it was described that, at the time when the process formalisation exercise commenced, a suitable model for describing interactions between

engineering process activities could not readily be sourced from the available literature, and that it was therefore problematic to formulate an appropriate reference framework for the organisation of the engineering activities.

In section 3.4.2.4 it was concluded that the AFCS development process was challenged by the conundrum of a "first article" engineering approach vs. an engineering approach that uses the conventional series of prototypes for development.

In section 3.4.2.4 it was also indicated that a root cause of the observed qualification problems was the utilisation of an inappropriate lifecycle model for the development and qualification of a sizeable volume of embedded software.

It was identified in section 3.4.2.4 that the lack of a process description which is commensurate with the development of a first article, without the use of prototypes, as well as being commensurate with the processes applied by Armscor, lead to difficulties in executing the Oryx navigation system upgrade programme. In particular, the problem of adequately handling feedback, i.e. when requirements or implementations of requirements must change as a consequence of e.g. integration or test results, was highlighted.

A significant number of lifecycle models which address these problems, representing different points of view and applicable to different situations, are described in the literature. The research challenge stemming from these observations is therefore:

> *An appropriate lifecycle model for the development of airborne electronic equipment, in particular resolving the problem of information feedback, is not determined.*

3.5.4 Development process controls

It was shown in the retrospective assessments that ineffective technical control over the development process negatively impacted the efficiency by which the projects described in this chapter were executed.

In section 3.3.3 it was pointed out that the control mechanisms to manage the traceability between data items and to ensure the correctness and integrity of the data produced during the development process was insufficient in terms of meeting with airworthiness requirements.

A cause of the flawed development strategy, with regards to the development of the Rooivalk AFCS (as described in section 3.4.2.4), was that there was no appropriate control mechanism to ensure correct execution of defined engineering activities.

In section 3.4.2.4, it was concluded that the management of the lifecycle data during the development of the AFCS was ineffective and caused inefficiencies as well as certification difficulties.

In section 3.4.3.4, it was concluded that a lack of a suitable baseline control mechanism had detrimental effects on the efficiency by which the programme was executed.

The fourth research challenge is therefore:

The development process control mechanisms are ineffective.

3.6 Research purpose / goal

The overall research challenge, in summarising these research challenges, is thus to mitigate the inefficiencies brought about by the lack of a well-defined and sufficiently comprehensive frame of reference for the planning of the projects, as well as ineffective control over the execution of the tasks comprising the projects.

This corroborates the purpose of the research, as stated in Chapter 1, which is re-iterated below:

Synthesise and validate a process to ensure cost effective development of platform specific airborne electronic equipment in the South African industrial and military environment.

Table 1: Research problem validation

Research Challenges / Information source	The scope of the engineering process for the development of airborne electronic equipment is not pertinently established.	The activities and tasks required for the development of platform specific airborne electronic systems are not identified explicitly.	An appropriate lifecycle model for the development of airborne electronic equipment, in particular resolving the problem of information feedback, is not determined.	The development process control mechanisms are ineffective.
Development process evolution assessment	x	x	x	x
AFCS development assessment: Development strategy		x		x
AFCS development assessment: Software development			x	
AFCS development assessment: Qualification and airworthiness certification		x	x	
AFCS development assessment: Change management				x
Oryx navigation system development assessment: Development process model			x	
Navigation system development assessment: HMI definition		x		
Navigation system development assessment: Integration and test				x

This table above shows the research challenges addressed by this study, derived from the initial problem statement. In the literature study (Chapter 4) this data is used in a traceability matrix to map literature study topics to research challenges.

3.7 Summary

In this chapter, typical attributes of airborne electronic equipment were presented first. This was followed by a discussion of engineering process concepts concerning the development of airborne systems. The research problem was analysed by means of observations from projects that were executed by Denel Aviation. This information was used to define shortfalls in the existing engineering processes as they existed at Denel Aviation. These shortfalls were discussed and translated to research challenges, in completing the requirements aspect of the relevance cycle of the DSR method. The research challenges were summarised in a traceability matrix in Table 1 and provide inputs to the literature study, where the research problem is further analysed and potential solutions are sought from available literature.

Chapter 4
Literature study

4.1 Overview

This chapter comprises the rigour cycle of the DSR method. Subject matter literature, i.e. the formalised knowledge base applicable to this study, is reviewed against the research challenges formulated in Chapter 3.7. The flow of information is shown in Figure 13. This illustration indicates how information from relevant research areas is used to transform research challenges into solutions regarding the stated research problem, by means of deductive and inductive reasoning. This chapter concludes with a matrix that shows how each research area addresses the defined research challenges, and how solutions to these research challenges are constructed.

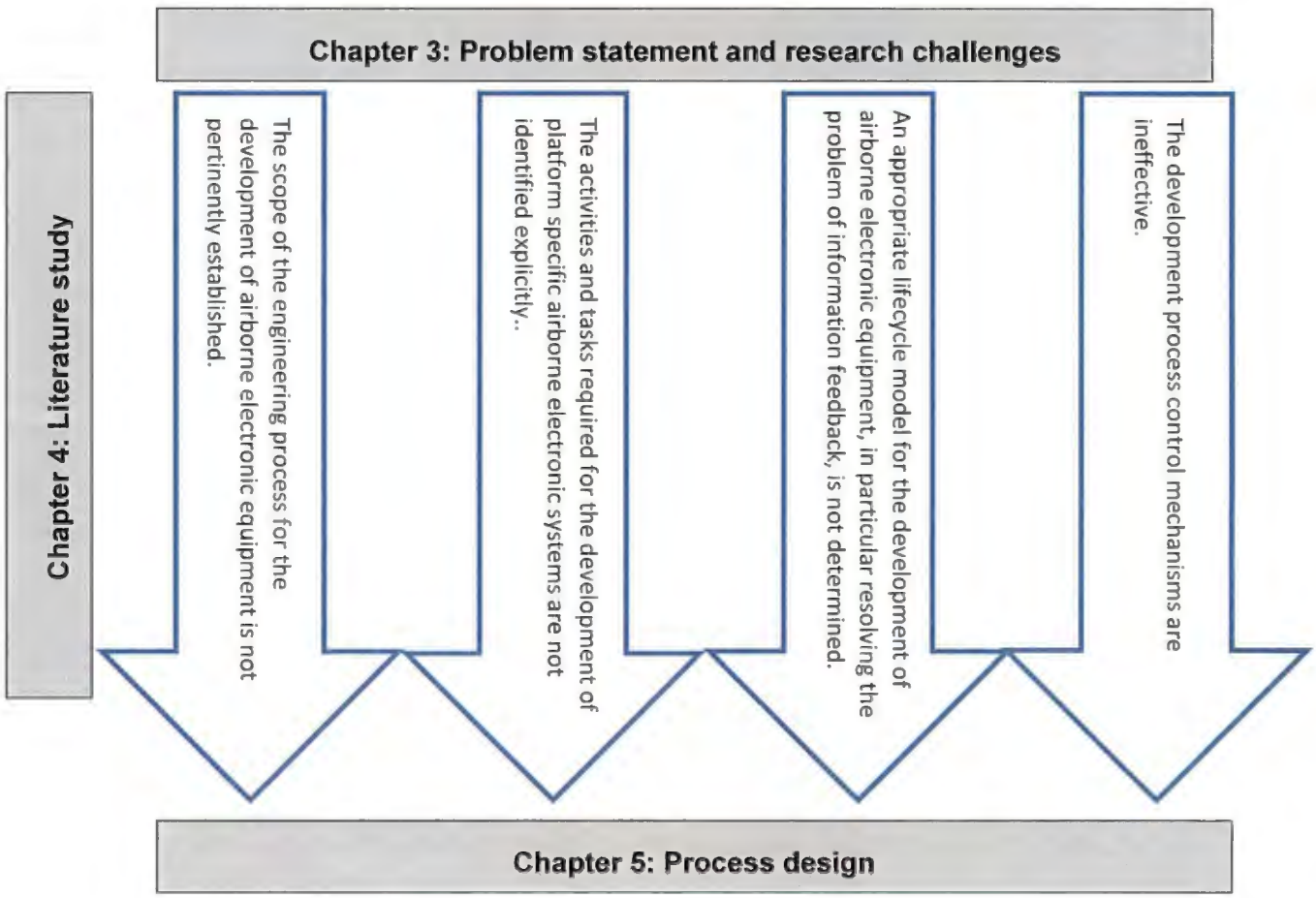

Figure 13: Literature study topics and focus areas

4.2 Introduction

4.2.1 The notion of systems engineering

Although the concept of systems engineering is a recent development in the realm of engineering practise, it must be recognised that systems engineering principles were applied in some way or another throughout the history of civilisation. Although not formally managed as such, these principles included the identification of a specific need, the identification of appropriate technologies to implement the intended solution, the physical realisation of the desired product and the eventual testing of the completed product to confirm that it was suited for its intended purpose. Systems engineering as a concept comprises a formalisation of these notions in a structured manner.

Furthermore, the systems engineering approach provides project planners with tools to adopt a holistic approach to the problem at hand from the outset of the project. A well-defined approach helps to identify all aspects that need to be addressed during the conception, production and operational use of the system-of-interest or product.

The formalisation of systems engineering principles can be traced to just after the Second World War. An article by Kenneth J Schlager - Systems engineering: key to modern development [22] - was published in the July 1956 edition of IRE Transactions on Engineering Management and provided the first formal description of the concept of systems engineering. The first formal systems engineering standard was MIL-STD-499 [16], published in 1969. Since then a plethora of standards and other publications saw the light, presenting a considerable resource for the accumulation of the collective systems engineering body of knowledge. In addition, systems engineering concepts are studied in the academic domain and by professional societies such as the International Council on Systems Engineering (INCOSE).

A sobering view on the dogmatic adoption of systems engineering standards and principles is presented in a 2010 paper by Joseph E. Kasser [23], where he highlights several shortcomings in the presently acknowledged systems engineering standards and viewpoints. He summarises his findings as follows:

> "...systems engineering is currently a discipline characterised by debates based on subjective opinions, with participants talking past each other, a lack of listening and a number of myths."

An internet search on topics such as systems engineering and requirements management also reveals that a significant number of opinions and notions regarding the engineering of a system currently exists. It transpires that the approach to the

problem depends largely on the type of system and the intended environment in which the system is to be utilised, and that the most suitable concepts as presented in the literature need to be adopted, tailored and implemented by the organisation engaged in the development process.

4.2.2 Organisation of the literature study

In order to provide structure to the topics related to systems engineering and other relevant topics studied in this chapter, the literature review is organised under the following headings:

- Classical systems engineering approaches;
- Contemporary systems engineering standards;
- Quality management processes and standards;
- Configuration management practises;
- Airworthiness recommended practises;
- Academic literature.

In the presentation of this literature study, the development process aspects described in each referenced information source is explained concisely to underline essential aspects that are relevant to this study. For every reference or group of references, as applicable, the described material is assessed in terms of the research challenges, under the headings:

- Scope of engineering activities;
- Definition of tasks and activities;
- Development process framework;
- Development process controls.

These headings correspond with the four research challenges formulated in Chapter 3.

Please note that, except where indicated to the contrary, all figures presented in the literature study are own contributions by the author, i.e. the figures are interpretations of the subject matter portrayed by the figures.

In order to guide the reader through the literature study, a roadmap of the major items reviewed in this chapter, as well as the motivation for its inclusion in the literature study, are presented in Table 2.

Table 2: Roadmap for the literature study

Literature group	Items reviewed	Reason for inclusion in the study
Classical references	• Blanchard & Fabrycky • RSA-MIL-STD-3 • MIL-STD-498	• Widely referenced in defence industry • Legacy with respect to applied processes
Contemporary systems and software engineering standards	• IEE 1220 • ISO/IEC 15288 • ISO/IEC 12207 • ANSI/EIA 632	• Widely used for guidance in planning of present-day development projects
Quality management standards	• ISO 9000/1 • SAE AS9100	• Organisations developing airborne systems must comply with these standards
Configuration management standards	• ISO 10007 • EIA-649	• Widely established methodology used in military and aerospace industries
Airworthiness recommended practises	• SAE ARP4754 • SAE ARP4761 • RTCA/DO-178B/C • RTCA/DO-254	• Establishes airworthiness principles to be incorporated in process design
Academic literature	• INCOSE handbook • SEBoK • Journal articles	• Provide further perspectives on engineering process lifecycles and principles

The literature review is followed by a summary of the solutions to the research challenges, which consolidates findings from the literature study.

4.3 Classical systems engineering approaches

Although the classical systems engineering methodology has generally been superseded by more modern approaches, some of the principles are still entrenched in the processes used by military acquisition organisations and therefore these methods are considered below. Furthermore, these methodologies laid some groundwork for contemporary processes, therefore a review of these classical

approaches provide further background to a better understanding of contemporary systems engineering views.

4.3.1 The classical approach

In the systems engineering model, as described by Blanchard and Fabrycky [44], the activities constituting the process are classified according to the lifecycle phases of Conceptual Design, Preliminary Design, Detail Design and Development, Production and Operational Use, and System Support, shown in Figure 14.

According to this model, the initial identification of the need, analysis of the user's requirements, the evaluation of feasible technologies for incorporation in the system, and system and program planning should be accomplished in the Conceptual Design Phase. The engineering plans, notably the SEMP, TEMP, Quality Assurance Plan (QAP) and Configuration Management Plan (CMP) and the top layer system specification (Type A specification) is presented to the client or contracting agency at a Conceptual Design Review. The configured documentation constitutes the Functional Baseline (FBL). An XDM prototype is built to aid in the process of determining requirements and to evaluate the feasibility of the concept.

In the next phase, the Preliminary Design Phase, the requirements emanating from the Conceptual Design Phase are subjected to further analysis and refinement, design trade-off studies are conducted and a preliminary system design is developed, system requirements are allocated to subsystems, and lower tier sub-contracts are placed. An ADM prototype is built to test design concepts, and to validate design decisions. Outcomes of this development phase include the development specifications, process or product specifications and material specifications. which forms the basis for the Allocated Baseline (ABL) for presentation to the stakeholders at the Preliminary Design Review (PDR).

This phase is followed by the Detail Design and Development Phase, where subsystems and components are designed, prototypes are developed, manufacturing and production processes are verified and developmental tests and evaluations are performed. The production planning process is also initiated during this stage. An EDM is developed to test the suitability of the system in an operational environment. The outcomes of this process include a set of detailed specifications that forms the Product Baseline (PBL). The Critical Design Review (CDR) is used to communicate the status of this baseline to all concerned parties.

CONCEPTUAL DESIGN	PRELIMINARY DESIGN	DETAIL DESIGN AND DEVELOPMENT	PRODUCTION	OPERATIONAL USE AND SYSTEM SUPPORT

B/L	FUNCTIONAL BASELINE	ALLOCATED BASELINE	PRODUCT BASELINE	UPDATED PRODUCT BASELINE
Plans	SEMP TEMP			
Specs	A-Specifications	B-, C-, D-, E-Specifications	C-, D-, E-Specifications	
Reviews	CER	PDR	CDR	

Figure 14: Systems acquisition process activities

Full scale production is commenced in the Production/Construction Phase and the system is eventually handed over to the user, where the system is deployed operationally, in a lifecycle phase designated the Operational Use and System Support Phase. When appropriate, the Production/Construction phase can be preceded by a Commissioning Phase, where the system is employed and evaluated operationally. The configuration of the deployed and accepted system forms the Operational Baseline (OBL). During the Construction/Production phase it needs to be demonstrated that the system meets the stated requirement as defined in the user's requirements.

4.3.2 RSA-MIL-STD-3

Development of military equipment in South Africa is normally accomplished under the auspices of Armscor, the procurement agency for the Department of Defence. Armscor is governed by the Public Finance Management Act of 1999, and manages acquisition programs according to internal processes which ensure compliance to the act. It is therefore evident that when a contractor provides a product according to an agreement between Armscor and the contractor, it is expected from the contractor to align their processes with Armscor's internal development control process. At the time of publication of this study, Armscor invokes RSA-MIL-STD-3: Programme Baseline Standards [15], as a reference for defining project milestones and activities.

4.3.2.1 System decomposition and hierarchies

In the classical systems engineering model, represented in this instance by RSA-MIL-STD-3 [15], a large military system is decomposed in sub-layers of hierarchy, or "System Levels", as shown in Table 3. The standard recognises the fact that the elements of a system at layer N become systems at layer N-1, e.g. layer 5 became systems at layer 4. The systems discussed in this study exist at system layer 3, 4 and 5 of this definition.

Table 3: Military system decomposition strategy

System Level	Type	Example
8	Operational Force	Air Force
7	Combat Grouping	Helicopter Systems
6	User System	Medium Transport Helicopters
5	Products System	Navigation
4	Product	Navigation Computer
3	Product Subsystem	Navigation Computer Software
2	Components	Connectors
1	Materials / Processes	Conformal coating of printed circuit boards

The specification structure shown in Figure 14 is associated with this hierarchy. The top level requirements (level 5 for the type of systems considered in the study) are recorded in the A-Specification, which defines the system as functional and identifies design constraints. This is a so-called "black box" specification, i.e. it contains no details about a proposed solution. In response to the A-specification, a B1-specification or "Prime Item Development Specification" is produced. The B1-specification is a "white box" specification, i.e. it contains a description of the system architecture and major components, as well as a designation of the allocation of system functions to system major components. A type C-specification defines the physical details of the system or subsystem. On lower layers, a type D-specification describes detail manufacturing processes and a type E-specification specifies material requirements.

4.3.2.2 Process description

RSA-MIL-STD-3 embodied the systems engineering approach promoted by the US military standards that dominated the military industry in the latter part of the 20th century. In this approach, the system lifecycle is divided into an Acquisition Phase and an Operational Phase. A graphic representation of the Acquisition Phase as described by RSA-MIL-STD-3 is shown in Figure 15. The description that follows is concerned with the Acquisition Phase, as it relates to development process aspects.

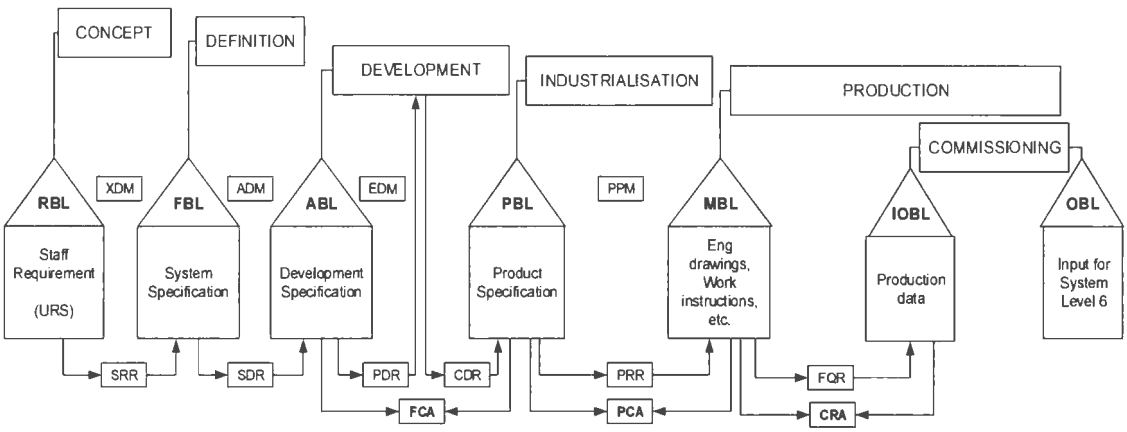

Figure 15: RSA-MIL-STD-3 development process

The Acquisition Phase process is characterised by the use of specific project sub-phases, particular baselines, reviews and audits, and the construction of distinctly defined prototypes. The Acquisition Phase is divided into Concept, Definition, Development, Industrialisation, Manufacturing and Commissioning sub-phases. Each of these sub-phases is associated with a baseline. Each baseline serves to consolidate and document the results of activities performed during a specific sub-phase. Note that for contracting purposes, the (technical) baselines are also associated with project milestones.

Not shown in Figure 15, but inferred from the context, is the first stage of the project that is performed by the end user, i.e. a Functional Study. The key outcome of the Functional Study is a User Requirements Statement (URS), which includes the user's functional and logistic requirements, as well as other project objectives, e.g. quantities and deployment targets. At the end of the Concept sub-phase, after conducting a formal User Requirements Review (URR), a Requirements Baseline (RBL) is established. The data constituting the RBL form the input to the Concept sub-phase. The functional and performance requirements are developed during this phase, based on the URS. One or several XDMs may be developed to test the practicality of concepts

originating during this sub-phase. The key outcome of this sub-phase is the System Specification and associated documentation, which forms the FBL. The lifecycle data constituting the FBL is approved after the System Requirements Review (SRR) was conducted.

The Concept sub-phase is succeeded by the Definition sub-phase, as per RSA-MIL-STD 3, of which the objective is "to allocate requirements to individual subsystems, product or items, thus establishing development objectives for each such subsystem, product or item". A task not stated explicitly in the standard, but required as a matter of consequence, is the development of a conceptual design in terms of a system architecture showing major subsystems and interfaces. The requirements developed during the concept phase are allocated to the subsystems, providing for the detailed specification of subsystem characteristics. ADMs can be developed to test whether the stated performance parameters can be achieved. The Definition sub-phase is concluded by conducting a PDR and establishing the ABL, which includes the Development Specification and Integrated Support Plan.

During the Design Development sub-phase, the system and its constituent elements are designed, developed and qualified in terms of the Development Specification and Integrated Support Plan. Typically, an EDM is constructed to demonstrate form, fit, function and tactical capability. The results of the Design Development sub-phase are captured by the PBL, which is reviewed at a CDR. The final design data of the system is consolidated in a Product Specification. A Functional Configuration Audit (FCA) is conducted at the end of the development cycle on all configuration items (CI) constituting the system, as well as on the comprehensive system, to validate that the CI has achieved the functional and performance characteristics allocated to it. The findings of the audit are captured in an FCA report. In practical terms, the FCA report reflects results of tests on the EDM representation.

Following the CDR and the establishment of the PBL, the Industrialisation sub-phase is entered, in which the manufacturing process is developed and qualified. A Pre-Production Model (PPM) that fully represents the final product is produced by using the manufacturing processes and equipment. It must be shown that the product and product systems can be manufactured at an acceptable standard and production rate. The outcomes of the Industrialisation sub-phase are consolidated and documented in the Manufacturing Baseline (MBL), and are reviewed at the Production Readiness Review (PRR). The PPM is subjected to a Physical Configuration Audit (PCA) to verify that the system, as built, complies with the Product Specification.

Full scale production of the Production Models (PM) is performed during the Production sub-phase. The data items that document the results of the production sub-phase, e.g. test and quality audit results, are reviewed at the Formal Qualification Review and form the Initial Operational Baseline (IOBL). A Commissioning Readiness Audit (CRA) is conducted on the production data to ensure compliance with the MBL data.

The objective of the Commissioning sub-phase is to integrate the deployment of the system into the user's operations. The Commissioning sub-phase typically includes an Operational Test and Evaluation (OT&E) program, where the capability of the system is evaluated against the URS. After completion of the Commissioning sub-phase the OBL is established and the system is deployed operationally.

The formal reviews and audits are conducted according to the guidelines of MIL-STD-1521 [9]. Note that although this standard was cancelled in 1995, the standard provides details for the understanding of the classical baseline centred model.

4.3.3 Synopsis of RSA-MIL-STD-3

4.3.3.1 Scope of engineering activities

The "classical" systems engineering process, as represented by RSA-MIL-STD-3, describes the comprehensive development effort and does not differentiate between e.g. technical processes and project processes as described in the contemporary systems engineering standards, which are discussed in Section 4.4.

This corroborates the statement of the first research challenge, i.e. the scope of the engineering aspects of the development process is not clearly delineated.

In terms of the second aspect of the process scope, as formulated in the first research challenge, it is evident from the data presented in Table 3 that the RSA-MIL-STD-3 approach is characterised by a rigid break-down of the system into levels of the hierarchy, with a markedly different approach to the lifecycle at each level of the hierarchy, including the types of documents to be generated at each level. This differs from the contemporary view of a "system of systems" as described in e.g. EIA-632 [12] where similar paradigms are applied at all levels of the system hierarchy.

Reasoning for the adoption of the "system of systems" method promoted by contemporary systems engineering standards, e.g. EIA-632 [12] for the development of airborne electronic equipment rather than the classical method will be presented in section 4.4.9.

4.3.3.2 Definition of activities and tasks

RSA-MIL-STD-3 is a high-level document and does not describe the details of tasks and activities in a sufficient level of detail to be used in terms of guidance for the design of a process. A high level summary of the process is presented in Table 4.

Table 4: RSA-MIL-STD-3 Acquisition phase process summary

Sub phase	Major activities	Model	Baseline	Major outcomes	Reviews / audits
Concept	Analyse the URS and develop functional and performance requirements of the system. Test the practicality of concepts.	XDM	FBL	System Specification	SRR
Definition	Develop the conceptual design and allocate requirements to individual subsystems, products and items. Test if stated performance parameters can be achieved.	ADM	ABL	Development Specifications	SDR
Design Development	Design, develop (manufacture and integrate) and qualify the system and constituent elements in terms of fit, form, function and availability.	EDM	PBL	Product Specifications	PDR CDR FCA
Industrialisation	Develop (design and establish tools, equipment and procedures) and qualify the manufacturing process.	PPM	MBL	Engineering drawings, Works instructions	PRR PCA
Production	Manufacture / establish the materiel to be used for commissioning and operating the system in the operational environment.	PM	IOBL	Production records	FQR CRA
Commissioning	Set up the operational capability of the system. Perform operational test and evaluation against URS.	PM	OBL	OT&E reports	

4.3.3.3 Development process framework

An important element of the process described by RSA-MIL-STD-3 is that the lifecycle view presented by the standard compels the progression of the development cycle through the following activities in a strictly controlled sequential manner:

- Requirements definition;

- Development of a conceptual or high level design;
- Detailed design;
- Creation of elements in hardware and software;
- Integration;
- Testing;
- Industrialisation;
- Production;
- Commissioning;
- Deployment and use;
- Support;
- Decommissioning and disposal.

Transitions from one phase to the next are only permitted when the preceding phase is shown to be complete. The development process is characterised by a series of reviews and configuration audits, which are in turn associated with specific project milestones. Along the way, a series of prototypes is produced which supports the objectives of the process at specific stages.

It is important to see that in such a process, where transitions between stages are strictly controlled, it is assumed that lifecycle data established in one stage will not be affected by the results of consequent stages. However, the problem of processing information that only becomes evident at advanced stages of the project, while also affecting upstream data (e.g. verification results that necessitate changes to designs) was highlighted in the retrospective assessments and gave rise to the research challenges that relate to development process lifecycle models and control mechanisms.

In terms of the development of a product, i.e. not considering the industrialisation and subsequent sub-phases, the methodology promoted by RSA-MIL-STD-3 manages feedback of information that originates during different stages of the development process, as shown in Figure 16. Information acquired during the concept stage, e.g. resulting from constructing and testing of the XDM, is evaluated and incorporated in the documentation that defines the FBL. RSA-MIL-STD-3 implies that sufficient information is obtained from the development of the XDM and the consequent evaluation process that it can be presumed that the system specification will require

no updates prompted by downstream activities. Similarly, all evaluation loops are closed during the definition stage prior to establishment of the ABL, implying that the development specifications will require no changes necessitated by outcomes of the design development sub-stage. The design development sub-phase encompasses the detailed design, manufacture of an EDM, integration and verification testing actions. Although not stated explicitly, it is assumed that all integration problems will be resolved and that the design data will be updated accordingly before the final qualification tests are performed on the EDM.

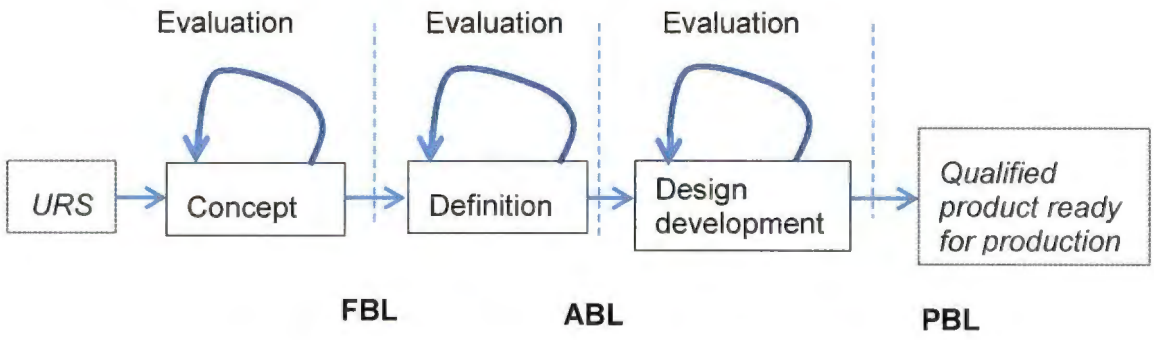

Figure 16: Management of development feedback

Therefore, although the approach represented by RSA-MIL-STD-3 can be classified as a pre-specified, single step lifecycle model (this classification is explained and elaborated upon in Chapter 4.8.2.2) due to the once-only transition through the sequence of baselines, it must be noted that the evaluation loops shown in Figure 16 implicitly contain elements of Iterative and Incremental Development (IID), which is described in Chapter 4.8.1.2. This mechanism enables the strict control of transitions between development stages, as prescribed by this standard.

The baseline structured development management process, described by RSA-MIL-STD-3, provides an efficient method of control and risk mitigation. It does, however, lean strongly on the premise that a sufficient effort is devoted to the development of the XDM and ADM prototypes that were developed with the aim of evaluating concepts, eliciting requirements, demonstrating functionality and verifying compliance with specifications. If the setting up of these prototypes do not form part of the development process, as was the case for the projects described in the retrospective assessments, then the stages, as defined by the reviews and configuration audits as required by RSA-MIL-STD-3, cannot be readily applied. Therefore, the lifecycle model described in this standard is not appropriate when a "first item" development process is followed.

It must further be noted that the fact that the classical systems engineering approach, represented by RSA-MIL-STD-3, is not appropriate for the development of systems utilising embedded software validates the research challenge that requires the identification of a lifecycle model appropriate for the development of airborne electronic equipment, in particular resolving the problem of information feedback.

4.3.3.4 Development process controls

An important observation is that the content of RSA-MIL-STD-3 was mainly developed in support of the interests of the acquirer. Development stages that are aligned with project milestones, where the acquirer could assess the technical development as well as the progress of the project against a schedule, are defined. These milestones are, in turn, associated with particular reviews and configuration audits. At the early stages of the evolution of the development processes at Denel Aviation, these reviews and audits were considered to be sufficient in terms of validation of the items subjected to the reviews. It was however realised that, due to the fact that these reviews took on the form of a forum discussion, the reviewers typically did not have sufficient time to study the material. It was also realised that the objectives of the reviews were twofold, i.e. assessments of technical correctness and the measuring of progress against the project schedule. The quality of the correctness of the data items could not be ascertained adequately to the levels required for airworthy equipment. This lead to the introduction of different methods for establishing the correctness and completeness of lifecycle data, as described in Chapter 4.9.4.

It can, therefore, be concluded that the control methods described by RSA-MIL-STD-3 could not be readily adopted for the development of airborne electronic equipment.

4.3.4 MIL-STD-498

MIL-STD-498: Software Development and Documentation [10] was released on 8 November 1994 and cancelled on 27 May 1998.

Although MIL-STD-498 was essentially a software development standard, it provided an extremely comprehensive description of a system development lifecycle where the system "contained" the software. An important aspect of this standard is the detailed descriptions that are provided for all the lifecycle data to be produced in the course of the development of a system and its software. These descriptions are formulated in terms of Data Item Descriptions (DIDs).

This standard is described and reviewed in detail below to provide background in terms of the development of a software intensive system, as well as to show why this standard was not forthrightly implemented by Denel Aviation.

The MIL-STD-498 system layer lifecycle is presented in Figure 17. The numbers in brackets refer to section headings as they appear in the standard. Note that all the figures shown in the description that follows, as well as grouping of activities, are the author's interpretations; the figures do not appear in the standard.

4.3.4.1 Preparation for development

The process defined in this standard commences with the project planning and oversight (5.1) task. The requirements needed to perform the activities that lead to the realisation of the system are identified and recorded in the development plans. The standard provides DIDs for a Software Development Plan (SDP), Software Test Plan (STP), Software Installation Plan (SIP) and Software Transition Plan (STrP). The plans are used to identify and describe the methods to be used during: the development processes; the resource requirements; the identification of relevant standards; and planned outputs and schedules. The plans are used to communicate project details to all stakeholders and are to be updated when factors addressed in the plans change. The standard requires that the development plans are approved by the acquirer.

The task of establishing a software development environment (5.2) includes the establishment, control and maintaining of the following elements which comprises a software development environment: the software engineering environment (5.2.1), which enables the requirements analysis, design and implementation processes; the software test environment (5.2.2), which enables the qualification and other testing of the software; and the software development library (5.2.3), which facilitates the orderly development and subsequent support of the software. Software development files (5.2.4) are used to record development information.

4.3.4.2 Development process

The actual development of the system commences with the systems requirements analysis (5.3) task, where the set of requirements to be met by the intended system is defined and also, importantly, the methods that will be used to demonstrate that each requirement is met. These details are recorded in the System Specification; interface requirements are captured in the Interface Requirements Specification. This same

process applies iteratively to the subsystems that are identified in the system design process, which follows on from the requirements process.

The requirements analysis task is followed by the system design (5.4) task. Decisions about the behaviour of the system and the selection of system components are formulated and recorded during this activity. It is important to note that design decisions are at the discretion of the developer, i.e. the acquirer is not supposed to influence these decisions. The acquirer only gets involved in design decisions when the decisions are "reverse engineered" into requirements (derived requirements) which then need to be approved in accordance with contractual agreements.

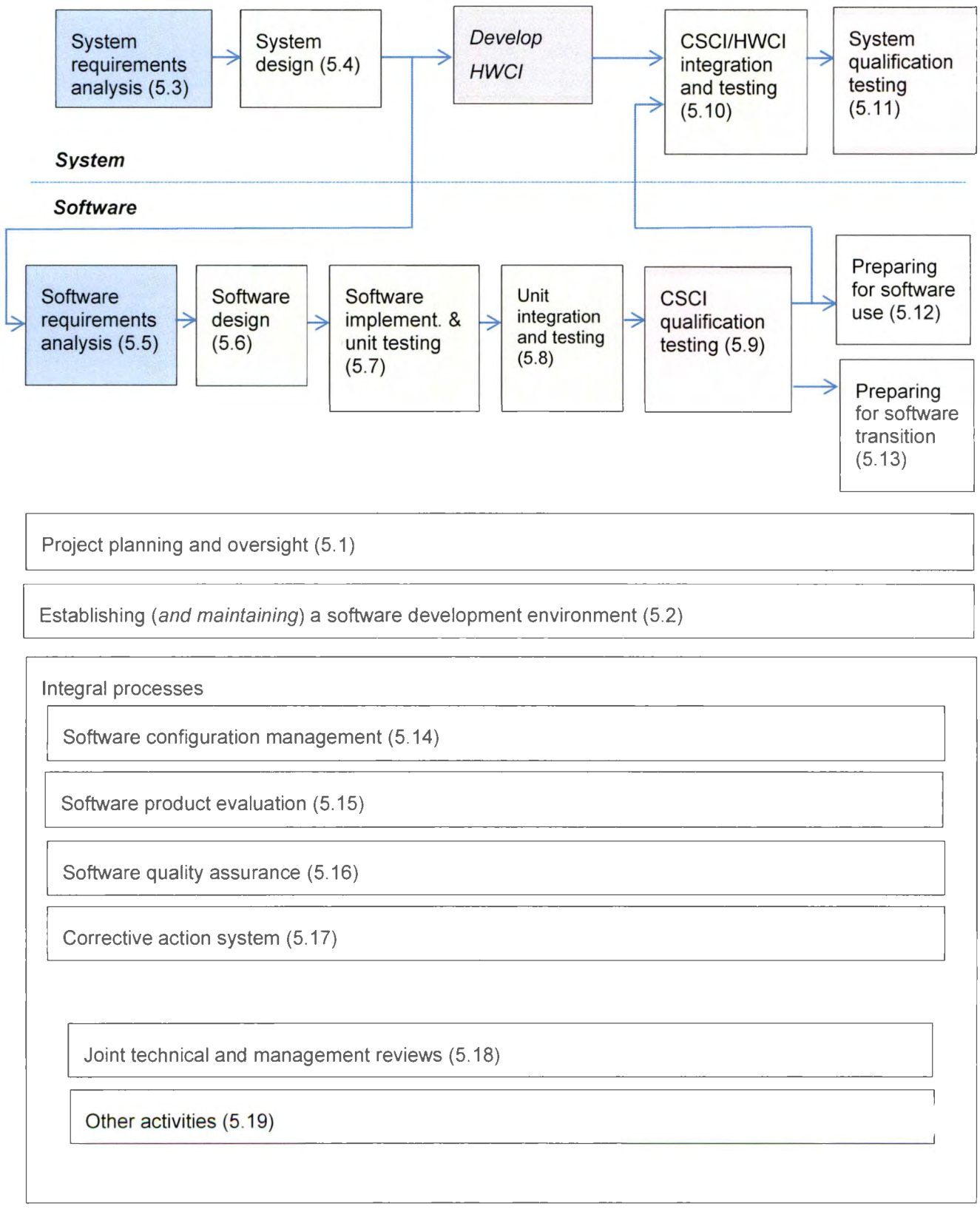

Figure 17: MIL-STD-498 development process elements

The developer needs to demonstrate the fulfilment of all these requirements by means of qualification testing. According to MIL-STD-498, the design decisions are viewed as developer-internal requirements to be implemented by the developer and imposed on subcontractors where applicable. The fulfilment of these requirements does not, however, have to be demonstrated to the acquirer.

The system architecture, that is the identification of system components and their interfaces as well as a description of interactions, is defined and recorded in the System/subsystem Design Description (SSDD). The data showing traceability between system components and the system requirements is also recorded in the SSDD. Interface design data is recorded in the Interface Design Description (IDD). If the system contains a database, then the description of the design of the database is documented in a Database Design Description (DBDD).

Not stated explicitly, but inferred from the context, is the development of the system hardware components. This process is shown as a shaded block in Figure 17, to augment the comprehensive view of the process as it is presented in this figure.

The software development process commences with the software requirements analysis (5.5) task. The software requirements that are to be met by the current software build increment are identified, and the data showing traceability to the system requirements is recorded. As for the case of system requirements, the methods to be used to demonstrate that each requirement is met must also be identified. These details are captured in the Software Requirements Specifications (SRS). Interface requirements are captured in the Interface Requirements Specifications (IRS).

In the course of the next task, software design (5.6), decisions about the behaviour of the Computer Software Configuration Item (CSCI), and the software components that constitute the CSCI, are formulated and recorded in a Software Design Description (SDD). Importantly, software design decisions are viewed as developer-internal requirements, which need to be implemented by the developer, and which may be imposed on subcontractors to the developer where applicable. Again, the fulfilment of these requirements does not have to be demonstrated to the acquirer. The software architecture, i.e. the identification of software components and their interfaces, as well as a description of interactions between the components, are captured in the SDD. The data showing traceability between software components and software requirements is also recorded in the SDD. A detailed description of each software unit comprising the CSCI is recorded in the SDD as well.

The next task identified by the standard is software implementation and unit testing (5.7). The software implementation and unit testing process for a development increment is shown diagrammatically in Figure 18. The software design is realised into software source code and databases, as applicable. MIL-STD-498requires that the acquirer approves the programming languages to be used.

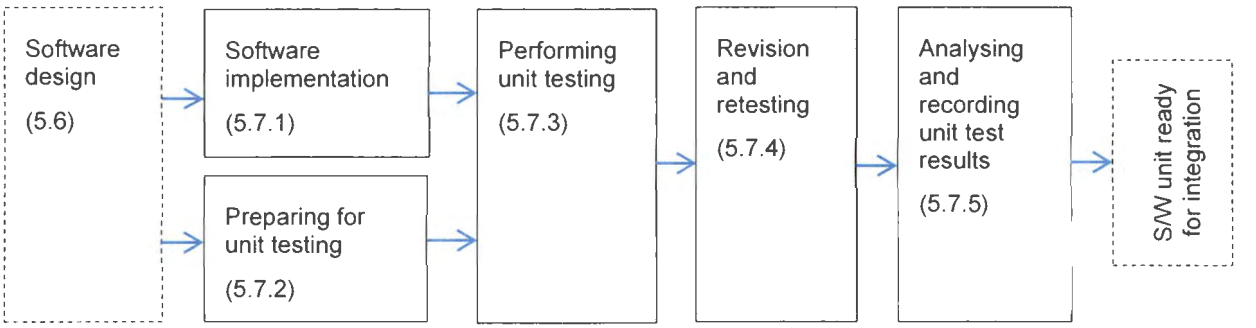

Figure 18: MIL-STD-498 software implementation and unit test lifecycle

The software units produced by the preceding activities are subjected to unit testing, that is, tests of compiled object code modules of individual units of source code in the host or target environment. Test cases that define inputs, expected results and evaluation criteria, test procedures and test data to be used during unit tests, are developed. The unit tests are then executed in accordance with the test cases and test procedures. Problems identified during unit testing are corrected and the updated units are retested. The standard requires that records of problems and updates are kept. After retesting is completed, the test results are analysed to confirm that all errors that were detected during the unit testing process, are removed. The results of these activities are recorded in Software Development Files (SDFs).

During unit integration and testing (5.8) the units emanating from the implementation and unit testing processes are integrated incrementally until all the units comprising the CSCI are integrated and tested. The software integration and testing process for a development increment is shown diagrammatically in Figure 19. Test cases defining inputs, expected results and evaluation criteria, test procedures and test data to be used during the software integration tests are developed. The test cases are to cover all aspects of the software design decisions, as well as the software architecture as they are described in the SDD. The software integration tests are executed in accordance with these test cases and test procedures. Problems identified during unit testing are corrected and the updated units are retested and records of problems and updates are kept. After retesting is completed, the test results are analysed to confirm

that all errors detected during the software integration testing process were removed. The results are recorded in SDFs.

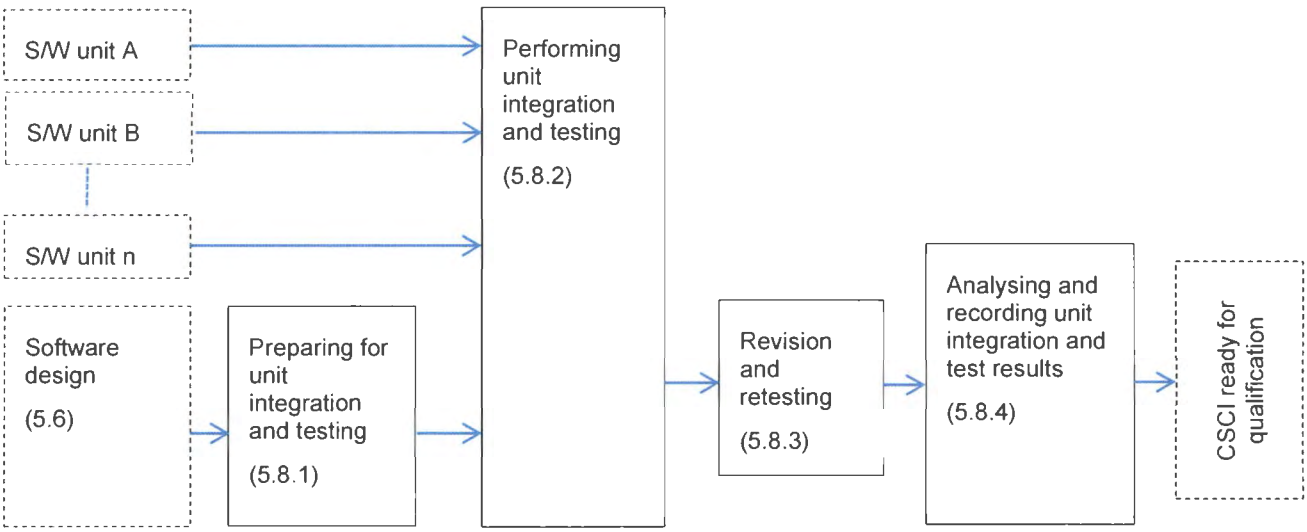

Figure 19: MIL-STD-498 software unit integration and testing lifecycle

The objective of CSCI qualification testing (5.9) is to demonstrate to the acquirer that the CSCI requirements, as recorded in the SRSs and IRSs, have been met. The standard specifies that the person or persons responsible for the detailed design or implementation of a CSCI are not those responsible for the qualification testing of the CSCI. Qualification testing is performed on the target computer platform, or on a platform agreed upon with the acquirer. The qualification testing process is shown in Figure 20.

Test cases defining inputs, expected results and evaluation criteria, test procedures and test data to be used during the CSCI qualification tests are developed, and data showing traceability between the test cases and CSCI requirements are recorded. These details are recorded as per the Software Test Description DID (DI-IPSC-81439).

The software qualification tests are then executed in accordance with the qualification test cases and test procedures, and are typically witnessed by a representative of the acquirer. Witnessed tests are preceded by dry runs of the qualification tests to ensure that the test documentation is correct and complete, and that the CSCI contains no errors. Problems identified during dry runs are corrected, the updated CSCI and test documentation are subjected to retests, and records of problems and updates are kept. Importantly, the standard provides for the instance where problems are identified during witnessed testing and allows for retesting. After qualification testing is

completed, the test results are analysed to confirm that the objectives of qualification testing are achieved. The results are recorded in the Software Test Report (STR), as per the STR DID (DI-IPSC-81440).

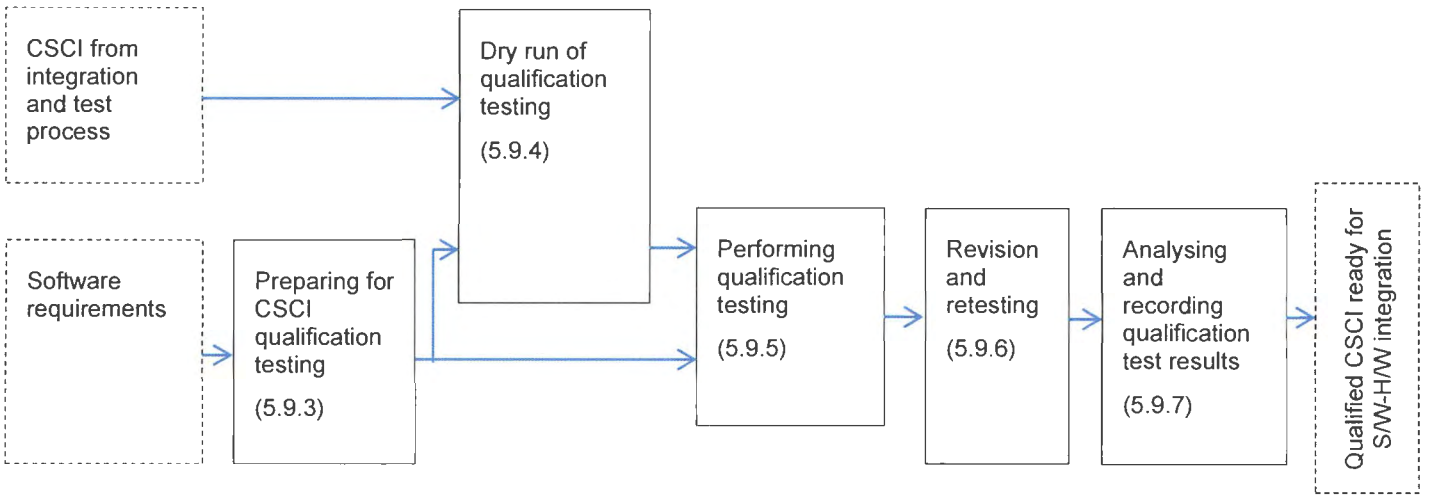

Figure 20: MIL-STD-498 CSCI qualification test lifecycle

During the CSCI/Hardware Configuration Item (HWCI) integration and testing (5.10) process, the CSCI is integrated with the final computer hardware system that is to be delivered to the acquirer. The CSCI/HWCI integration and testing process is a developer-internal activity. This testing process concludes with high level system tests.

The CSCI/HWCI integration and testing process for a development increment is shown diagrammatically in Figure 21. Test cases that define inputs, expected results and evaluation criteria, test procedures and test data to be used during the CSCI/HWCI integration tests, are developed. The test cases are to cover all aspects of the system and architectural design as described in the SSDD. The CSCI/HWCI integration tests are executed in accordance with these specified test cases and test procedures. Problems identified during CSCI/HWCI testing are corrected and the updated system is retested and records are kept of problems and updates. After retesting is completed, the test results are analysed to confirm that all errors detected during the CSCI/HWCI integration and testing process were removed. The results are recorded in SDFs.

The objective of system qualification testing (5.11) testing is to demonstrate to the acquirer that all the system requirements have been met. As in the case of CSCI qualification testing, MIL-STD-498specifies that the person or persons responsible for the detailed design or implementation of a CSCI are not to be those responsible for the qualification testing of the system. System qualification testing is performed on the target computer platform, or on a test platform agreed upon with the acquirer. The system qualification testing process is shown in Figure 22.

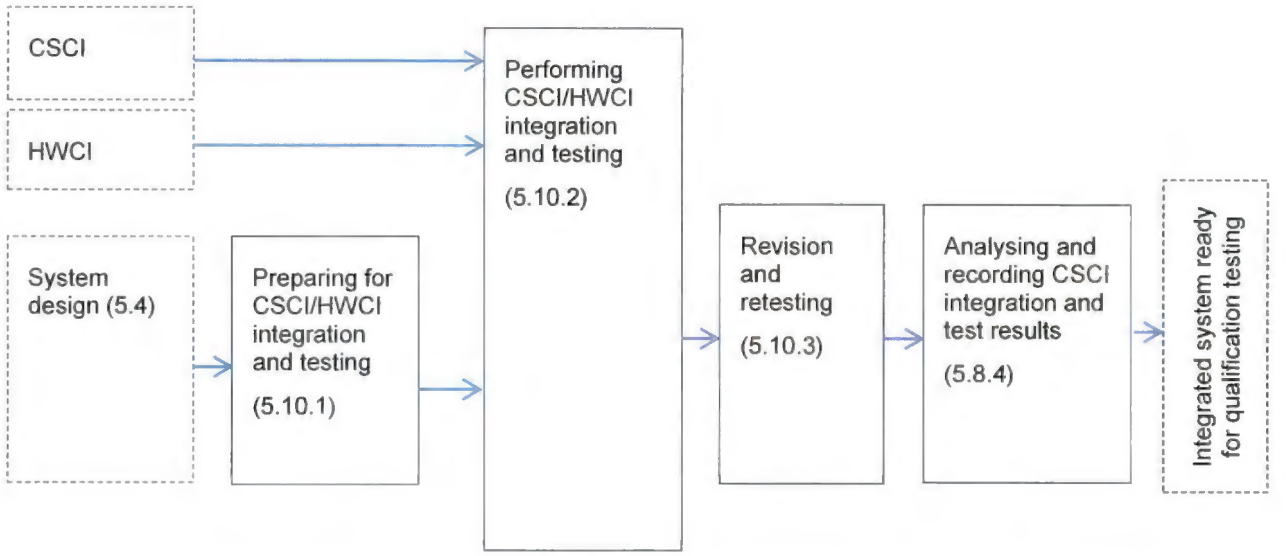

Figure 21: MIL-STD-498 CSCI/HWCI integration and testing lifecycle

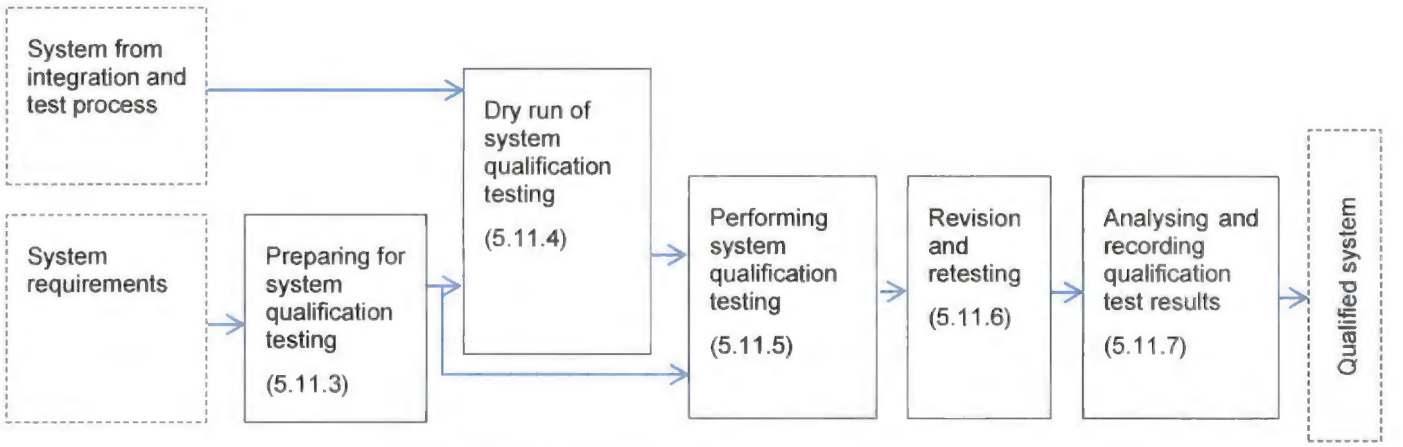

Figure 22: MIL-STD-498 system qualification test lifecycle

Test cases that define inputs, expected results and evaluation criteria, test procedures and test data to be used during the CSCI qualification tests are developed, and traceability between the test cases and CSCI requirements are recorded. These details are recorded as per the Software Test Description DID (DI-IPSC-81439).

The software qualification tests are executed in accordance with the qualification test cases and test procedures and are typically witnessed by a representative of the acquirer. These witnessed tests are preceded by dry runs of the qualification tests to ensure that the test documentation is correct and complete, and that the CSCI under test contains no errors. Problems identified during dry runs are corrected, the updated CSCI and test documentation are subjected to retests, and records of problems and

updates are kept . The standard again provides for the instance where problems are identified during witnessed testing and allows for retesting of corrected test items. After qualification testing is completed, the test results are analysed to confirm that the objectives of qualification testing are achieved. The results of this process are recorded in the STR as per the STR DID (DI-IPSC-81440).

The activities Preparing for software use (5.12) and Preparing for software transition (5.13) that are described in the standard do not apply to embedded systems, the development of an embedded system after completion of the qualification test lifecycle.

4.3.4.3 Integral processes

MIL-STD-498 refers to the following activities as "integral processes". These processes are applied throughout the development lifecycle.

Software configuration management (5.14) is discussed under the topics of configuration identification (5.14.1), configuration control (5.14.2), configuration status accounting (5.14.3), configuration audits (5.14.4) and packaging, storage, handling, and delivery (5.14.5) in the standard. These concepts are described in Chapter 4.6 of this dissertation, under the topic of configuration management.

In addition to qualification (5.11), the standard describes software product evaluation (5.15) tasks. The developer is required to perform in-process evaluations of software products. A final evaluation of each deliverable software product is to be completed before delivery, and to retain records of evaluations for the duration of the contract. It is also specified that the persons conducting the evaluations should not be the same persons who produced the items evaluated. The standard provides comprehensive guidance on procedures for conducting evaluations (Appendix D of the standard).

Software quality assurance (5.16) consists of evaluations of development activities and the resulting software products that should be conducted on an on-going basis to assure that activities are performed in accordance with the approved plans, and to assure that that the product has undergone evaluations, testing and corrective action as required by the standard. Records of quality assurance activities are to be retained, and it is also required that the quality assurance personnel are independent of the development personnel. Quality assurance forms part of the discipline of quality management, which is as a separate literature review topic, presented in Chapter 0.

The developer shall implement a corrective action system (5.17) using problem reports to record problems in software products under configuration control, ensuring that action is taken, resolution is achieved, status is tracked and records of the problem are

maintained. The standard provides further guidance on the corrective action methodology (Appendix C of the standard).

The standard requires that the developer schedules and conduct joint technical and management reviews (5.18) in conjunction with the acquirer, on technical and managerial matters.

In addition, the standard requires that the following other activities (5.19) are performed:

- Risk management (5.19.1): The software management plan should indicate the methods for the management of technical, cost, or schedule risk.
- Software management indicators (5.19.2): The methods by which the progress on software can be measured and reported must be identified. The handling of the software management indicators is to be defined in the Software Development Plan.
- Security and privacy (5.19.3): The developer needs to identify how security and privacy requirements, as defined in the contract, will be met, and how these will affect the development program and software products.
- Subcontractor management (5.19.4): It must be ensured that all the requirements relevant to the developer are also made applicable to subcontractors if they are used.
- Interaction with software IV&V agents (5.19.5): The developer must interface with Independent Verification and Validation (IV&V) agents, as specified in the contract.
- Co-ordination with associate developers (5.19.6): The developer must interface with other development groups as specified in the contract.
- Improvement of project processes (5.19.7): The developer must assess the effectiveness of the development process used on the project on a continual basis, identify opportunities for improvement and implement these improvements by means of approved updates to the software development plan. Note that process improvement also forms part of the discipline of quality management, presented in Chapter 0.

4.3.4.4 Development strategies

Appendix G of the standard provides guidance on program strategies, tailoring and build planning. Three types of strategies are identified, i.e. a "grand design strategy", an "incremental strategy", and an "evolutionary strategy".

The grand design approach is described as a "once-through, do-each-step-once" strategy. This strategy is essentially the waterfall model described in Chapter 4.8.3.2, strictly following the sequence:

- Determine user needs;
- Define requirements;
- Design the system;
- Implement the system;
- Test;
- Fix;
- Deliver.

In this strategy, all requirements are defined up-front; there is only one development cycle and no interim delivery of software.

The incremental approach follows the "grand design" approach up to the completion of the requirements definition. Thereafter, the rest of the development is completed in a sequence of builds. System capability is added in each incremental build until the completion of the project. Interim delivery of software may be considered.

The evolutionary approach allows for user needs and system requirements to be partially defined up-front, and these are refined during each build. The incremental builds are delivered to the user and employed as permitted by the level of functionality.

4.3.5 Synopsis of MIL-STD-498

MIL-STD-498 is assessed below, concerning its applicability to the stated research challenges.

4.3.5.1 Scope of engineering activities

The standard comprehensively defines the activities involved in the development of a system. However, it does not differentiate between e.g. technical processes and

project processes as described in the contemporary systems engineering standards, discussed in Chapter 4.4.

A principle introduced by this standard that is aligned with the contemporary system break-down view (which is described in more detail in Chapter 4.4 but which differs from the RSA-MIL-STD-3 method) can be inferred from the diagram presented in Figure 17, i.e. the same types of activities are performed on the system layer as on the software layer. This means that, on an abstract level, the scope of the development process at the system layer is the same as the scope at lower layers of the hierarchy.

The recursive use of the process at different layers of the system hierarchy will be adopted in the process design.

4.3.5.2 Definition of activities and tasks

The standard is document-based and very prescriptive, down to low level details. A comprehensive set of DIDs accompanies the standard, which details the contents of every lifecycle item to be produced during the development process. When contracting against this standard, compliance to each paragraph in the applicable DID must be shown, which significantly constrains developers in terms of latitude of technical descriptions.

Modern development processes rely more and more on tools that provide for the electronic capture, manipulation and management of data. Therefore, this highly documentation based process is not appropriate anymore, and modern standards do not subscribe to the approach of prescribing the format and detailing the contents of documents. As an example, RTCA/DO-178C [25] does not prescribe lifecycle data formats, but requires that standards for lifecycle data types, e.g. requirements and design standards, are developed and that it must be shown that the lifecycle data produced during the execution of a project conform to these standards. This approach is discussed in detail in Chapter 4.7.4.

It must be noted that the standard presents and extensive definition of the tasks constituting the development process. However, it does not present clear associations with the system safety process, which is a necessity for airborne electronic equipment.

Importantly, the definition of specific activities, as well as the details specified in the DIDs, as presented by MIL-STD-498, provide suitable references for the identification of the activities that should be incorporated in the process design, described in Chapter 5.

4.3.5.3 Development process framework

MIL-STD-498 describes a development lifecycle which is graphically represented in Figure 17. This lifecycle resembles lifecycle models described in contemporary systems engineering standards, discussed in Chapter 4.4, and in the airworthiness recommended practises, described in Chapter 4.7. Specific observations regarding the lifecycle model described in MIL-STD-498 are described below.

The software/system integration, test and qualification process, as well as the test and integration iterations, as presented by the standard, are shown in Figure 23. In this development process a newly developed software unit is subjected to unit tests to confirm that the unit correctly performs its intended function. If problems are detected, the unit is reworked (the design and/or source code is modified) and retested. The unit test results are recorded and analysed, and this cycle continues until the unit passes the unit tests.

The units constituting a CSCI are integrated and subjected to integration tests. As in the case of unit tests, if problems are detected during this stage, the CSCI is reworked and retested, continuing the cycle until the CSCI contains no errors that can be detected at this stage. CSCI test results are analysed and recorded throughout this cycle.

After the completion of the CSCI integration activities, the CSCI is subjected to a dry run of the CSCI qualification tests with the objective to ensure that the CSCI test cases and procedures are complete and accurate, and that the CSCI is ready for witnessed tests. Updates to test documentation and/or software are to be incorporated as part of the dry run exercise. When the developer is satisfied that the witnessed CSCI qualification tests can be completed successfully, the formal CSCI qualification tests are conducted.

The standard indicates that, even though a dry run was performed, provision must be made for revision and retesting during the qualification tests. The qualified CSCI is then integrated onto the target hardware and tested to ensure that the CSCI executes correctly on the target platform. In this instance as well, a dry run of the system qualification test is performed prior to the witnessed tests.

The revision and retest loops do not necessarily relate only to the preceding activity – in many cases it may require alterations to the software unit itself, implying a return to the beginning of the process, with consequent updates to records throughout the cycle, up to the point where the problem was detected. This can place a large burden on the

baseline and version control process, and lead to excessive amounts of e.g. test data, where only the latest revision applies to the actual attributes of the unit or CSCI.

Note that, if the dry runs indicated in the description are performed at a sufficient level of rigour, the witnessed qualification tests should not require a revision and retesting cycle. The qualification and test lifecycle, as shown in Figure 22, can be redrawn as shown in Figure 24, without loss of generality, to demonstrate this scheme. An activity block was added to show the iterative loop between dry runs and making corrections. Another activity block is added to highlight the fact that all lifecycle data must be baselined before formal witnessed qualification tests can commence.

Note that these slight changes, based on the assumption that the objective of the witnessed qualification test is to demonstrate the compliance of a mature system, and that all non-compliances are removed through an iterative process prior to the qualification tests, result in a qualification lifecycle with minimal changes to the formally structured baseline.

The incorporation of dry runs of qualification tests in the integration process will be considered in the process design.

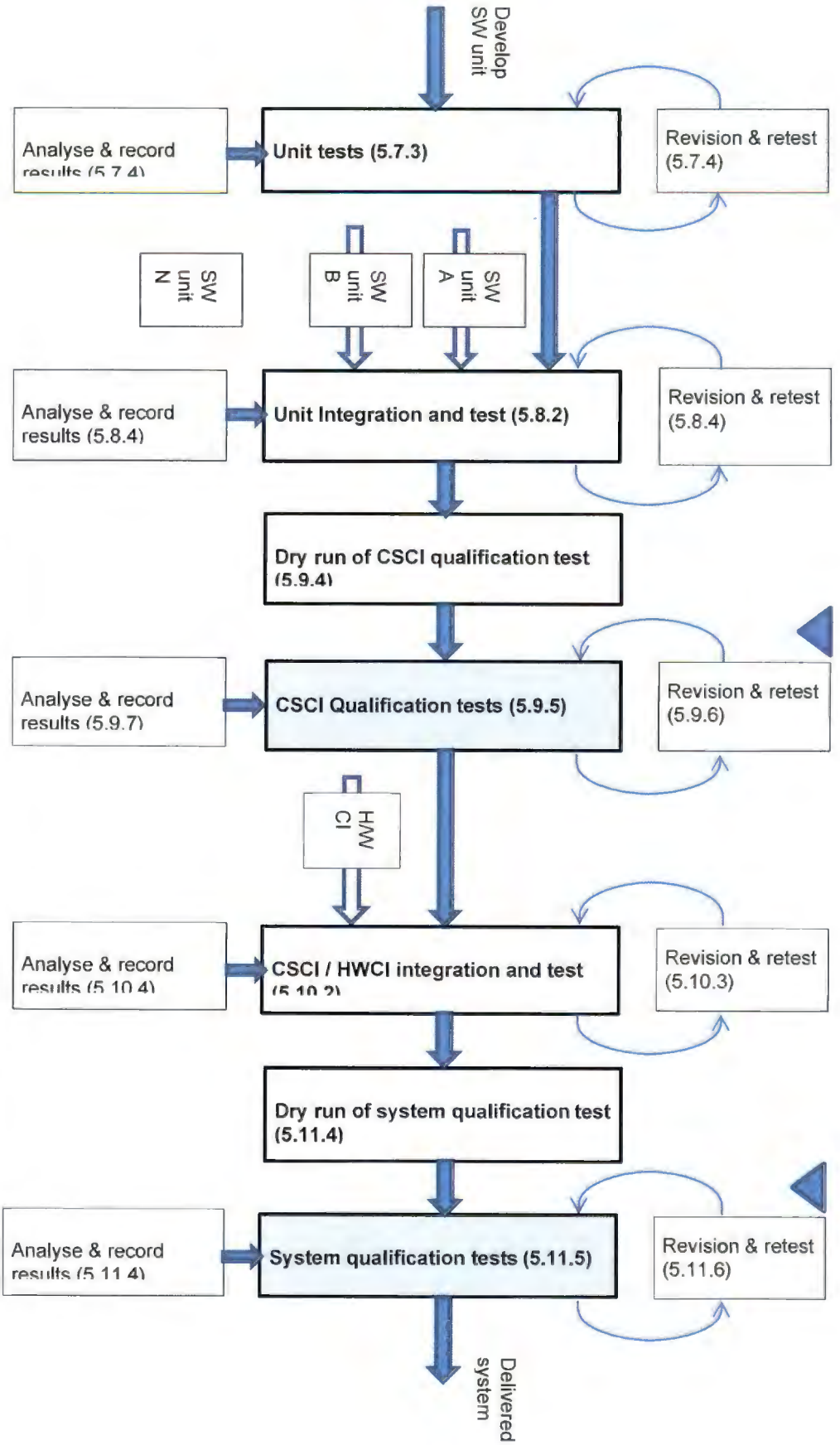

Figure 23: MIL-STD-498 test, integration and qualification cycle

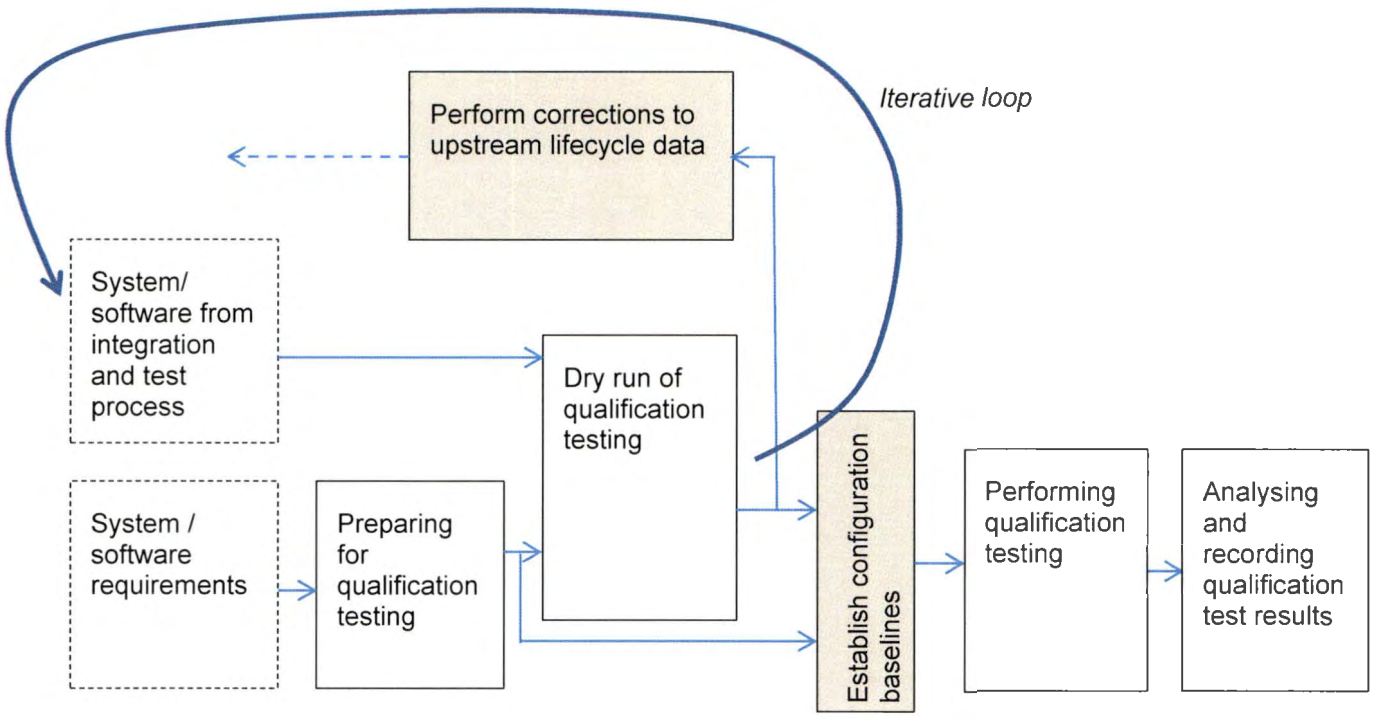

Figure 24: Modified qualification test lifecycle

4.3.5.4 Development process controls

A number of the "integral processes" described by MIL-STD-498 can be considered to be development control mechanisms. These are software configuration management, software product evaluation, software quality assurance, and the corrective action system. The joint technical and management reviews also serve as a control mechanism. Specific aspects of these mechanisms are discussed below.

- Software product evaluation

The process of qualification testing, which aims to demonstrate that the software product meets with its requirements and that it contains no errors, is accompanied by an in-process "software product evaluation" effort, as detailed in Appendix D of the standard. This evaluation is essentially a review, by independent assessors, of the items that constitute the software product, including plans, requirements data, design data, test documentation and test results.

- Corrective action system

The use of a corrective action system is an important element of a controlled development process and an aid to maintaining system integrity.

This aspect must be addressed in the process design.

- Joint technical and management reviews

The principle that the requirements (i.e. system or software specifications), as well as the proof of compliance to the requirements, must be approved by the acquirer, but that the implementation details are at the discretion of the contractor, is important. It allows the contractor to exploit applicable technologies and algorithms effectively and also protects the contractor's intellectual property associated with the design. It also absolves the acquirer from responsibility for design decisions, but assigns the responsibility for the final acceptance of the functional behaviour and performance of the system to the acquirer.

Therefore, this principle should be entrenched in the process design emanating from this study.

4.3.6 Classical systems engineering processes - conclusions

The classical systems engineering references, of which RSA-MIL-STD3 and MIL-STD-498 were typical examples, were included in the study due to their historical significance in the development of military equipment and the fact that Armscor, who is the contracting agency for the South African National Defence Force, still employs processes which were established according to these standards.

Salient aspects of the classical methods include:

- The standards were document-based and very prescriptive, down to low level details.
- The "military" systems engineering model represented a linear, sequential progression from definition, through development to use and support.
- The process included the development of a particular set of prototypes which were associated with specific decision points within the lifecycle. These decision points were controlled by means of well-structured reviews, associated with pre-defined baselines.
- System qualification is performed when design and integration is completed.

- A rigid break-down of the system in different levels of hierarchy, with a markedly different approach to the lifecycle at each level of hierarchy, including the types of documents to be generated at each level.
- The project schedule is based on a set of deliverables consisting of documents and formal reviews.
- There is no clear differentiation between project management and technical processes.
- MIL-STD-498 differentiates between evaluation activities and qualification.
- MIL-STD-498 identified the possibility of incremental development of the system, both in terms of development of requirements as well as in terms of delivery of functionality.

Aspects from the classical process description assessed above, which should be considered in the process design, include:

- The use of prototypes to support the development of requirements and to test implementation strategies, when it is appropriate;
- Use of the concept of baselines to control different stages of the development project, albeit in a different format than the baselines prescribed in the classical standards;
- MIL-STD-498 and MIL-STD-1521 are useful as checklists to confirm that specific tasks are considered in the process design.

The following aspects of the classical methods will not be incorporated in the process design:

- The method of basing the project schedule on a set of deliverables and milestones associated with specific documents and formal reviews;
- The method for differentiating between different classes of specifications at different layers of the system hierarchy;
- The use of prescribed prototype types at different stages of the lifecycle.

4.3.7 Classical systems engineering processes - summary

4.3.7.1 Summary of salient aspects

- The classical standards are highly prescriptive;
- The standards do not clearly separate project management and technical objectives;

- The processes defined in the standards are associated with the development of specific prototypes;
- The development process is characterised by specifically defined baselines, reviews and milestones;
- A hierarchical product structure is defined;
- A "Waterfall" development approach is promoted.

4.3.7.2 Summary of concepts applied / discarded

Apply:

- "Classical" standards referenced in development of checklists;
- Use standards in validation of process design.

Discard:

- No "classical" concepts included in process design.

4.4 Contemporary systems engineering standards

Systems engineering practises and conventional systems acquisition methods, in particular when associated with the development of embedded software, have been the subject of considerable study and rethink in recent years. These studies have culminated in a number of systems and software engineering standards that provide guidelines on processes concerned with engineering of systems. Prominent amongst these are EIA-632 [12], IEC 15288 [39], IEEE 1220 [38], and IEEE 12207 [46]. The practises advocated by these publications, which are all endorsed by authoritarian bodies in the respective fields, represent the culmination of experience and insight from a vast number of systems and software engineering practitioners.

It is therefore imperative that an establishment engaged in the development of modern complex systems take cognisance of industry best practises, and to integrate the applicable methods within its own processes and procedures.

When considering public domain engineering standards, it must be borne in mind that these are continuously subjected to revision and improvement. The standards described in the following paragraphs have undergone changes during the period covered by this study (1994 to 2015) and are referenced due to their significance with

respect to the development of airborne electronic equipment in the South African context.

4.4.1 ANSI/EIA-632

ANSI/EIA-632-1999: *Processes for Engineering a System* [12] was developed as a joint project for the Electronic Industries Alliance (EIA) and the International Council on Systems Engineering (INCOSE) and was first published in January 1999. The standard provides guidance to an enterprise with respect to processes for engineering of a system, focussing on the technical (as opposed to commercial) aspects.

ANSI/EIA-632 identifies the process for the engineering of systems within the greater organisation, and describes interactions with other processes. Figure 25 shows a representation of the context of the "Process Groups for Engineering Systems". In this view, the enterprise (which executes the engineering process) exists in an external environment, which includes laws and regulations, legal liabilities, social responsibilities, technology base, labour pool, competing products, standards and specifications (national/international), and a public culture.

This enterprise environment comprises of policies and procedures, standards and specifications (corporate), guidelines, domain technologies and local culture.

The project environment operates within the enterprise environment and consists of directives and procedures, plans, tools, project reviews and metrics. For a specific project, the project support processes of project management and agreement support, and the process groups for engineering systems, comprising acquisition and supply, technical management, system design, product realisation and technical evaluation are identified.

The project environment is supported by enterprise support processes, including investment decisions, external agreements, infrastructure support, resource management, process management, production and field support.

The project supports the technical processes by means of the directives and procedures that are prepared by the project in order to guide the technical effort and by providing project management services such as "project integration, scope management, time management, cost management, quality management, communications management, risk management, and procurement management".

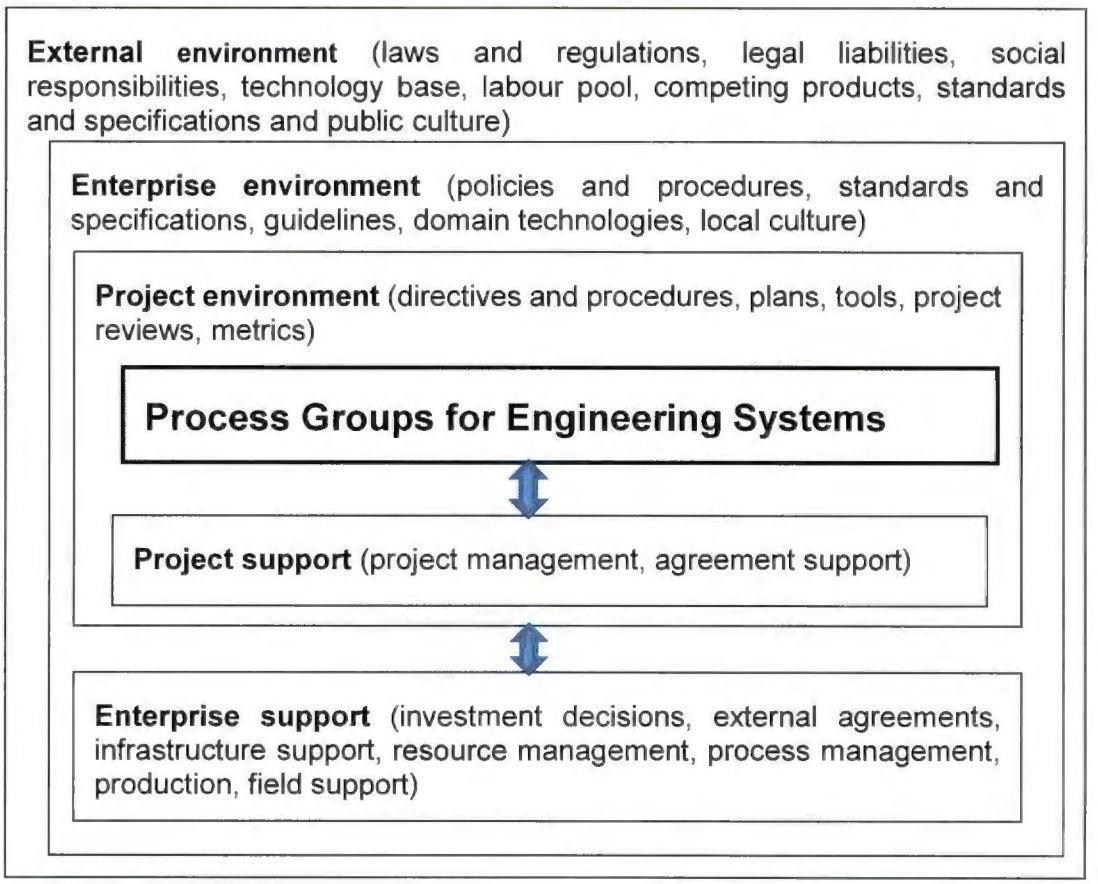

Figure 25: Systems Development Process Context, from ANSI/EIA-632

With regards to the system architecture, ANSI/EIA-632 defines the system as consisting of the end products, which are used for the intended purpose, and the enabling products that enable the creation and use of the end products, as shown in Figure 26. Enabling products include all the items required to develop, produce, deploy and support the end products. Examples of enabling products listed by the standard include plans, schedules, policies and procedures, automated tools, analytical and physical models, hardware and software development environments, mock-ups, test setups, training systems, maintenance record systems, etc.

The standard promotes the use of the concept of a generic building block, which is based on this system concept view, as shown in Figure 27. The building block consists of system elements, end product elements and enabling product elements. Each end product and enabling product typically includes hardware, software, firmware, personnel, facilities, data, materials, services, and/or processes. The building block provides the framework for the application of the processes. The building blocks are used by the standard for:

1. Identifying and assigning specifications for the system, end products, and subsystem elements;

2. Managing interfaces;
3. Enabling multidisciplinary teamwork;
4. Assessing risk;
5. Structuring technical reviews
6. Cost collection and reporting.

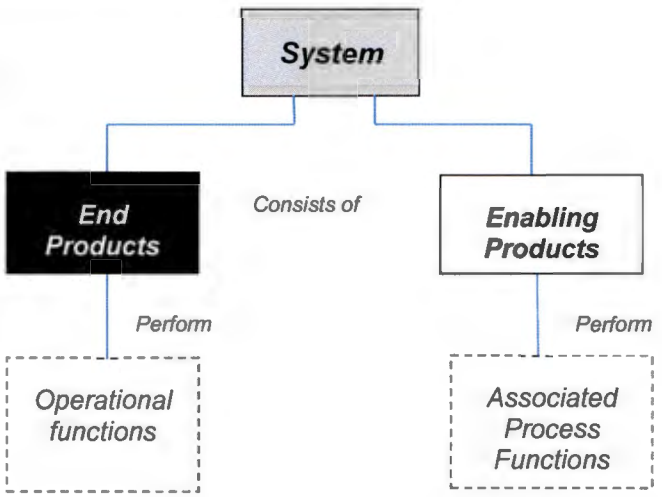

Figure 26: System concept (ANSI/EIA-632 [12])

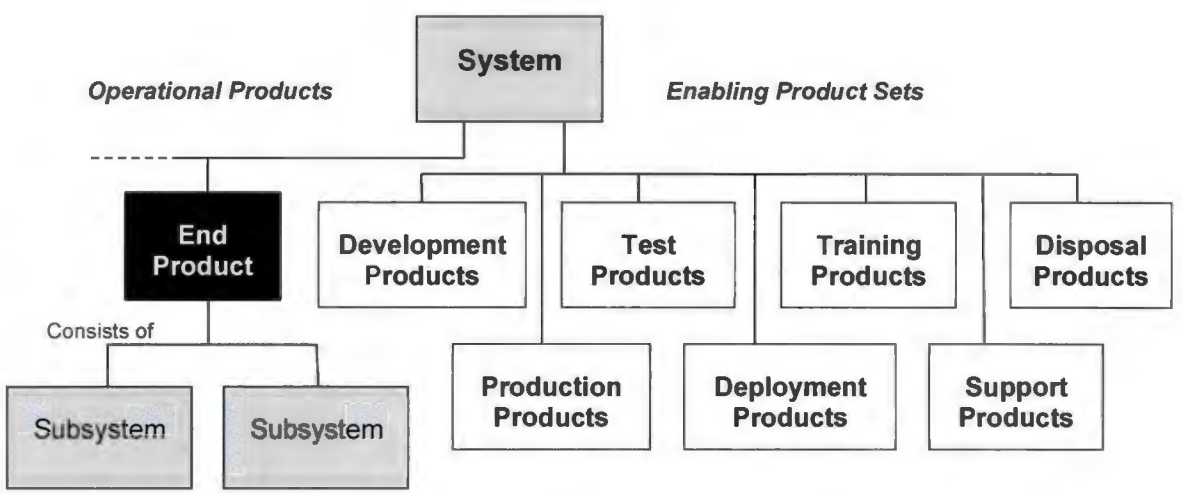

Figure 27: Building Block (ANSI/EIA-632 [12])

In the systems engineering model described by ANSI/EIA-632, requirements at a given layer of the system hierarchy are implemented via architecture and "black boxes" at the next layer, where each "black box" forms the requirements for this lower tier system architecture and "black boxes".

In order to realise the development of a complex system, the system-of-interest is viewed as a combination of a system along with its associated enabling products,

which are in turn each broken down into subsystems and their respective enabling systems. Each subsystem and enabling system is treated as a system on its own, with its own requirements, implementation and verification processes. These subsystems exist in a hierarchy defined by the system-of-interest architecture, and each subsystem is associated with its own architecture.

This concept of the system structure as described in the standard is shown schematically in Figure 28. Building blocks are connected to form the system structure or a building block hierarchy. Subsystems at one layer become systems at the next lower layer, and so forth. Not all subsystems are developed by the developer. Some may be sub-contracted or procured off-the shelf, but they can all be considered as separate building blocks conforming to the representation as described in Figure 28.

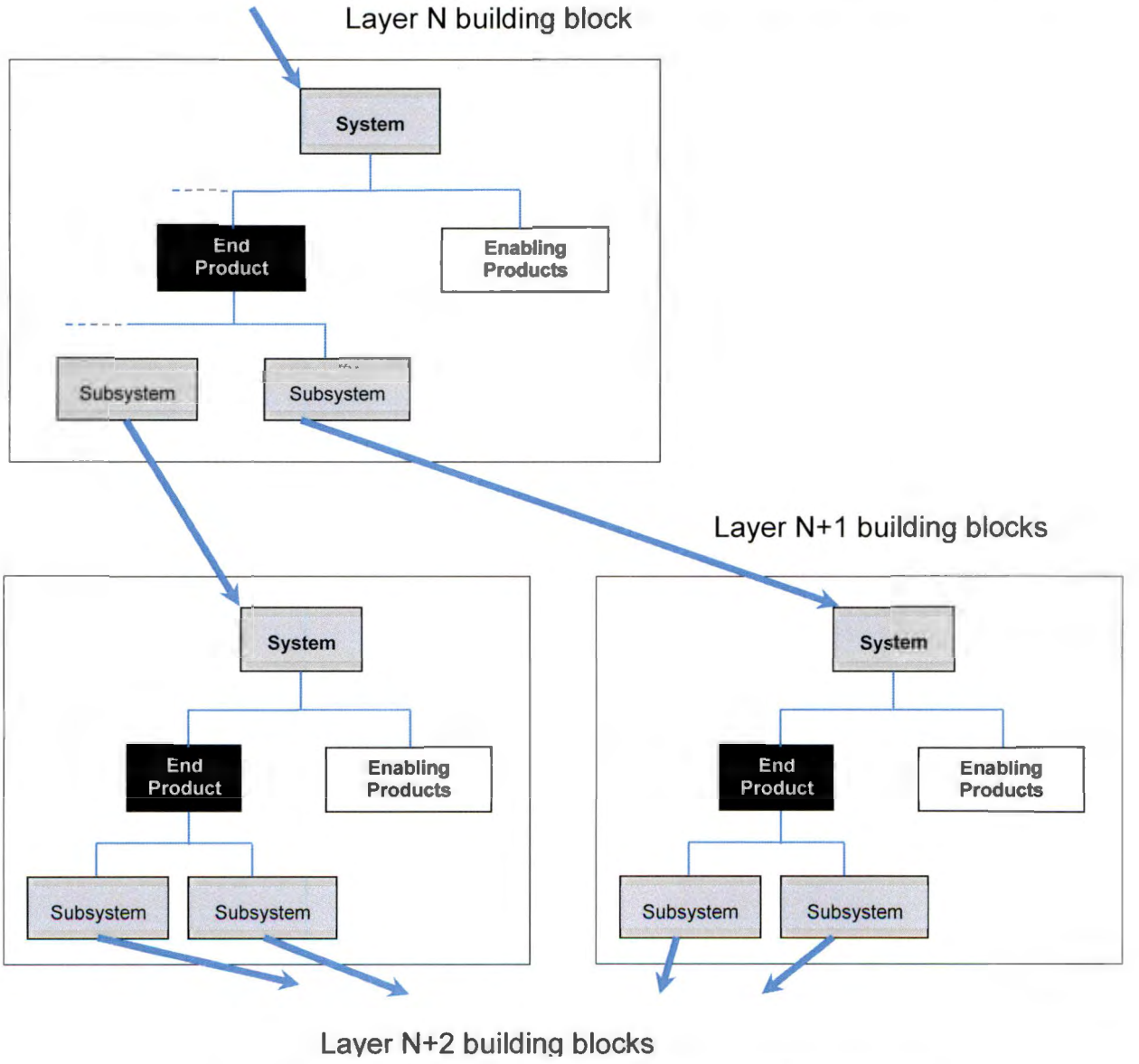

Figure 28: System structure concept (ANSI/EIA-632 [12])

It is also important to note that, according to the standard:

"[Specified] requirements for a subsystem become the assigned requirements at the next lower layer of development. Each building block can have other stakeholder requirements that are not related to the requirements that are either assigned from above or directed by users or customers."

This structure consequentially leads to "top-down" development, as the assigned requirements for a building block arise at a higher layer of system hierarchy.

Realisation of the system starts at the lowest layers of the system hierarchy, where each layer delivers verified end products to the next higher layer, and so forth up to Layer 1, in a "bottom-up" manner.

Importantly, at each layer, the end product specified requirements are used for the verification of the end product

Sub-clause 6.2.2 of the standard requires that the end product at a layer is verified against its specified requirements, and that the delivered product is then validated at the higher layer against its assigned specified requirements, as shown in Figure 29.

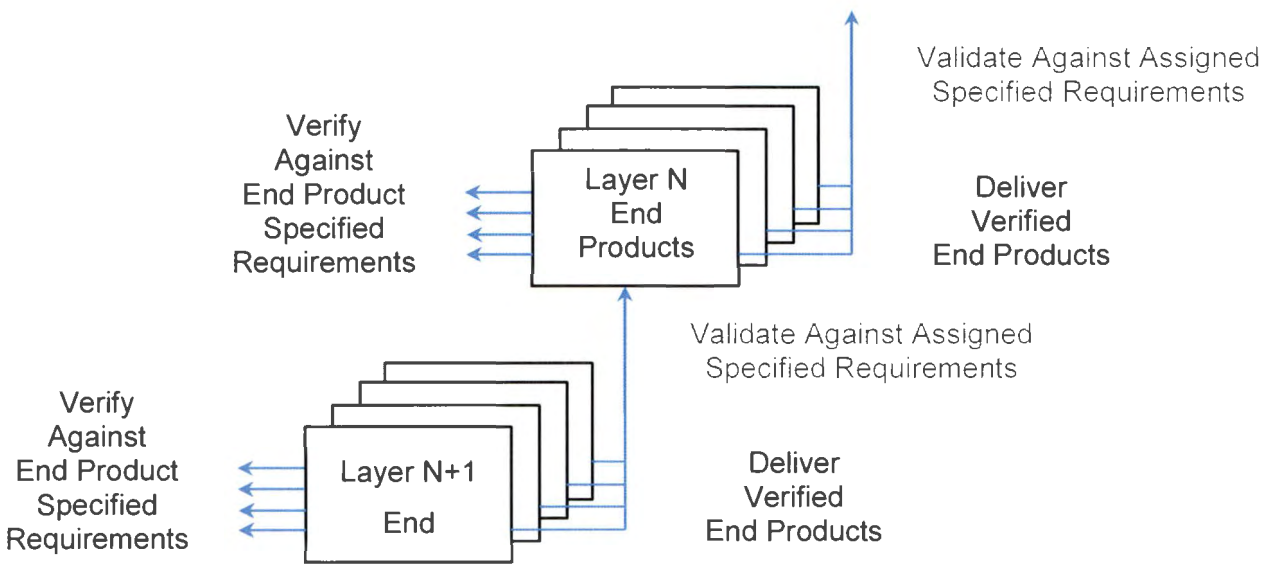

Figure 29: Bottom-up realisation (ANSI/EIA-632 [12])

ANSI/EIA-632 presents the following definitions of the term "lifecycle":

The engineering lifecycle is "[a] sequence of phases that evolves an instance of a system from a concept to a set of products consistent with the exit criteria established for an enterprise-based lifecycle phase". Sub-clause 6.3 of the standard describes the

engineering lifecycle in terms of lifecycle phases. These phases are shown in Figure 30.

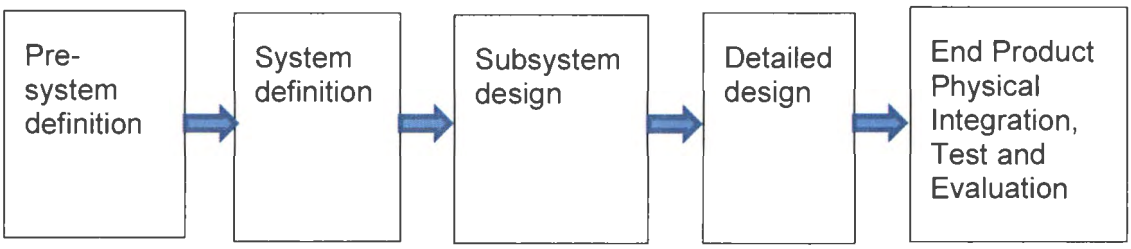

Figure 30: EIA 632 engineering lifecycle phases

Note that the enterprise-based lifecycle is defined as "the incremental progress of a system from conception through disposal, marked by management-established milestones with assigned exit criteria". ANSI/EIA-632 defines the enterprise as the "entity that has governance over a set of projects, or over organisations in which projects are carried out". This is the lifecycle of the system, from conception throughout its utilisation up to its final disposal. This lifecycle is not considered in this study, as the objective of this study is to define the process for the development of a product and its enabling products up to the stage of its release to operational service.

The standard describes a generalised systematic approach to the engineering of a system through the application of a set of processes to each element of the system, by an empowered team. This systematic approach is applicable to:

(1) Completing corrective actions;

(2) Making refinements;

(3) Developing derivatives;

(4) Producing modifications;

(5) Updating existing products;

(6) Creating and realising new systems;

(7) Disposal.

The standard indicates that "this approach is incrementally applied in an engineering lifecycle framework that can be implemented during any one or more phases of an enterprise based lifecycle".

Clause 4 of the standard lists 33 requirements for the processes used in engineering a system, organised into five groups: Acquisition and Supply; Technical Management;

System Design; Product Realisation and Technical Evaluation. Importantly, note that the groups and processes are neither hierarchical nor sequential, i.e. these requirements state "what" needs to be done; the sequence and detailed format is to be planned by the developer.

These requirements can be summarised in general terms as follows:

a) Acquisition and supply processes shall be in accordance with agreements (Requirements 1, 2 and 3);

b) The technical effort and schedule shall be planned and shall be in accordance with the agreement (Requirements 4, 5, 6 and 7);

c) Work shall be executed according to work directives, according to the plans and agreements, and the outcomes of the technical effort shall be managed (Requirement 8, 12);

d) The work progress shall be assessed and reviewed against plans and requirements (Requirement 9, 10 and 11);

e) Information shall be disseminated as required (Requirement 13);

f) Requirements management, including requirements validation, shall be implemented (Requirement 14, 15, 16, 19, 25, 26, 27 and 28);

g) A logical solution and physical solution shall be defined (Requirement 17 and 18);

h) The solution shall be implemented (Requirement 20);

i) A verified product shall be provided to the user (Requirement 21, 30 and 31);

j) Effectiveness and trade-off analyses shall be performed (Requirement 22 and 23);

k) Risk analyses shall be performed (Requirement 24);

l) Enabling products shall be assessed for readiness for its intended purpose and lifecycle (Requirement 32);

m) End products shall be validated for conformance to its validated requirements (Requirement 33).

For the purpose of this study, these requirements can be generalised and presented schematically as shown in Figure 31. The product development process follows a logical progression (not necessarily sequential) from requirements, to solution definition, to verification. The process is controlled against references (plans and

agreements) and major outcomes are validated. The solution definition process includes risk analyses and effectiveness and trade-off studies. This general structure provides a convenient basis for the assessment of the other process descriptions to follow.

Annex B of the standard presents an enterprise based lifecycle model which is linked to a decision making process. It identifies six typical phases, i.e. assessment of opportunities; investment decision; system concept development; subsystem concept development; subsystem design and pre-deployment; and finally, deployment, operations, support and disposal. The use of simulation and prototyping in support of the development process is also described in Annex B, as summarised in Table 5.

Table 5: Use of simulation and prototypes

Type	Support area
Simulation	Assessment of opportunities
Simulation, physical or functional prototypes	Investment decision phase
Advanced technology prototypes	System concept development
Pre-production prototypes	Subsystem design

Although it promotes a top-down layered approach, the standard recognises that three approaches can be followed to engineer a system, i.e. top-down, bottom-up, and middle-out. (Sub-clause 6.2.1). This means that the pre-system definition can be developed from user requirements (top-down), or from an existing system which are to be modernised (middle-out), or from an existing system to be reverse engineered (bottom-up).

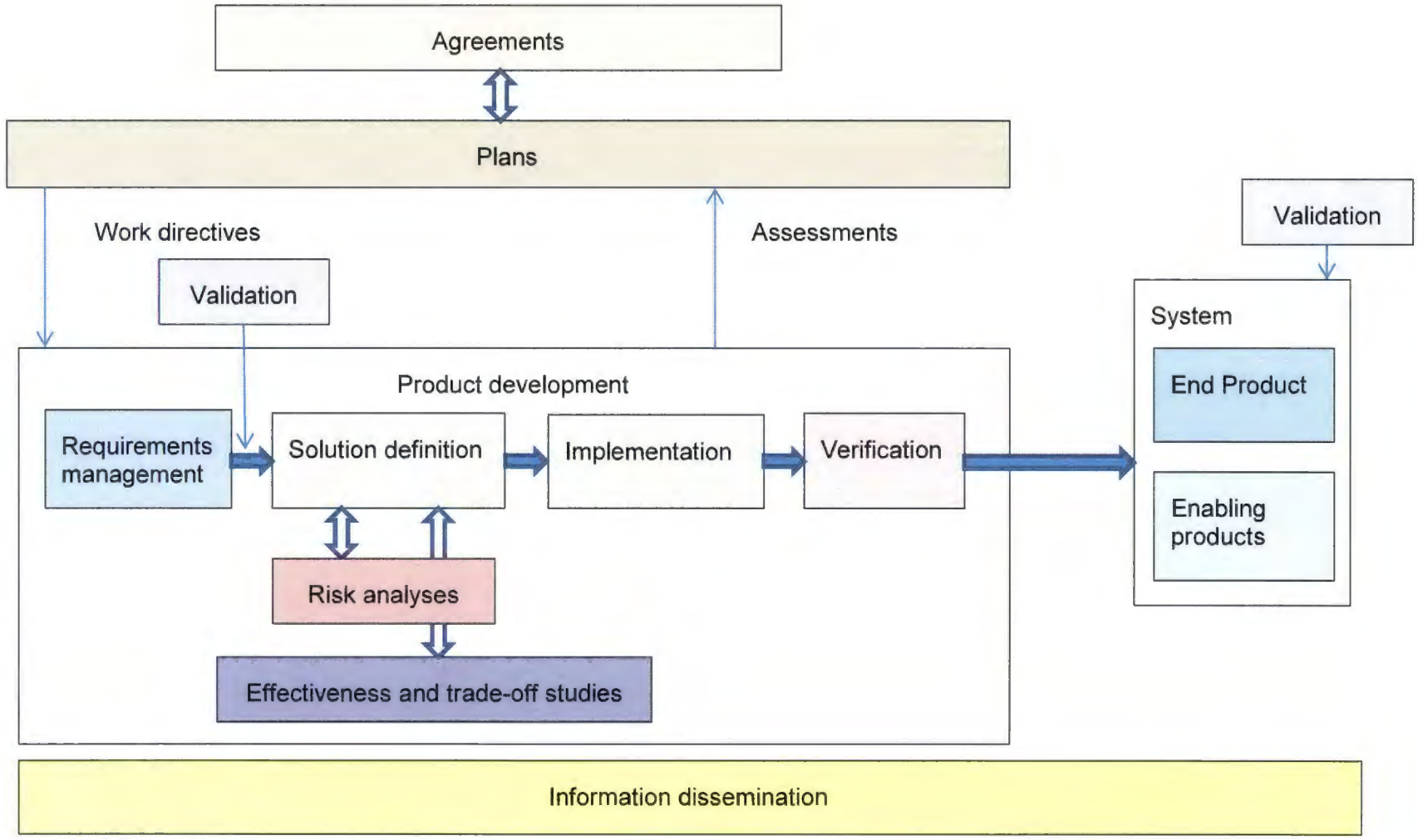

Figure 31: ANSI/EIA-632 engineering processes

4.4.2 Synopsis of ANSI/EIA-632

4.4.2.1 Scope of engineering activities

This standard outlines a well-defined partitioning of engineering tasks within the environment where they are executed, as shown in Figure 25. The presented view of the systems development context narrows down the scope of the development process, i.e. the tasks to be considered in a process design are isolated from other project and organisational tasks that require skills not associated with the technical aspects of the development of a system.

An important aspect related to the simplification of the scope of the development process is the method by which the standard breaks down the system architecture into end products and into enabling products, as shown in Figure 26 and Figure 27. This arrangement provides a convenient reference for the organisation of the process structure.

A further concept which is lucidly described by the standard is the notion that an end product in turn consist of subsystems, which in turn become systems-of-interest at a next tier of the system hierarchy, as shown in Figure 28. This implies that, at an abstract level, a generic development process can be designed for implementation at any layer of the systems hierarchy, which can be utilised throughout the system architecture.

These aspects of the process scope, i.e. a distinct partitioning of activities, a structure for organisation of the process and a generic development approach that can be applied at any layer within the system architectural hierarchy, are better defined and less involved than the aspects concerning the process scope as represented by the classical approach.

Therefore, in terms of the objectives of the DSR rigour cycle, the guidelines for defining the process scope, as presented in ANSI/EIA-632-1999: Processes for Engineering a System *[12], will be considered in the process design.*

4.4.2.2 Definition of activities and tasks

ANSI/EIA-632, by virtue of the 33 requirements that form the essence of the standard, presents a generalised depiction of the necessities to build a system, but not at the level of detail as presented by e.g. MIL-STD-498. Important observations with respect

to the activities that constitute a development program, described by this standard, are discussed below.

- Types of enabling products

Considering the enabling products depicted in Figure 27, it can be seen that some of the enabling products are developed to support the realisation of the system, e.g. development, test, and production products, whereas other enabling products support the system during operational use, e.g. training, deployment, and support products. The former can be considered as development enablers and the latter are typically treated as integrated logistic support elements.

This logical breakdown of enabling products provides a convenient and comprehensive framework of development process groups to which tasks of the development process can be allocated.

- Prototypes as enabling products

The standard describes the development of prototypes, along with the use of simulation in support of development processes, as shown in Table 5. Note that in the classical systems engineering model, as depicted in Table 4, prototypes are defined elements of specific lifecycle stages, whereas prototypes (and simulations) constitute enabling products in the development process described by ANSI/EIA 632, and that they can be developed according to a specific need, i.e. there is no classification of e.g. XDM, ADM, etc.

It will be shown in Chapter 4.6.7 that a certain level of rigour of control is required for the "core" elements of an airworthy system. The notion of prototypes and simulations as enablers, rather than being core elements of the system, hence requiring a lower level of rigour of control, allows for a tidy definition of the process architecture.

4.4.2.3 Development process framework

Annex B of the standard identifies six typical phases, i.e. assessment of opportunities; investment decision; system concept development; subsystem concept development; subsystem design and pre-deployment; and finally, deployment, operations, support and disposal. However, for the purpose of this study, only the stages from system concept definition to deployment are of interest. These can be viewed as to constitute the development process.

The product development lifecycle is shown in Figure 31. The development process as described by ANSI/EIA-632 can be classified into four major elements, i.e.

requirements management; solution definition; implementation; and verification. This group of activities can be implemented iteratively to produce the system end product and its enabling products. The similarity between this lifecycle model and the lifecycle models described in other contemporary systems engineering standards will be pointed out in the sections to follow. It is also important to note that the standard points out that the lifecycle model is applicable to top-down, bottom-up, and middle-out requirements implementation strategies.

This identification of the generic elements of requirements management, solution definition, implementation and verification provides a sound basis for the further research to determine a suitable lifecycle model.

4.4.2.4 Development process controls

The controls associated with the development process are also shown in Figure 31. Work is executed according to a set of project plans, and product development process outputs are assessed against these plans. The plans are in turn aligned with an overall agreement. This ensures that work will be executed in such a form that outputs are traceable to, and in compliance with the agreement that authorised the development project.

The principle of assessing product development outputs against formalised guidance and instructions, e.g. project plans approved by the acquirer, provides a means of ensuring consistency of project outputs and it reduces risks of non-compliance with customer expectations. This concept is important in determining the control mechanisms for the process presented in Chapter 5.

4.4.3 ISO/IEC 15288

ISO/IEC 15288: *Systems and software engineering – system lifecycle processes* [39] presents a "common framework for describing the lifecycle of systems created by humans". The scope of this standard is much wider than that of the standards discussed above, but it provides less detail with respect to the lifecycle processes.

The standard defines a system structure as shown in Figure 33, and the lifecycle processes described in the standard are applied in relation to this structure.

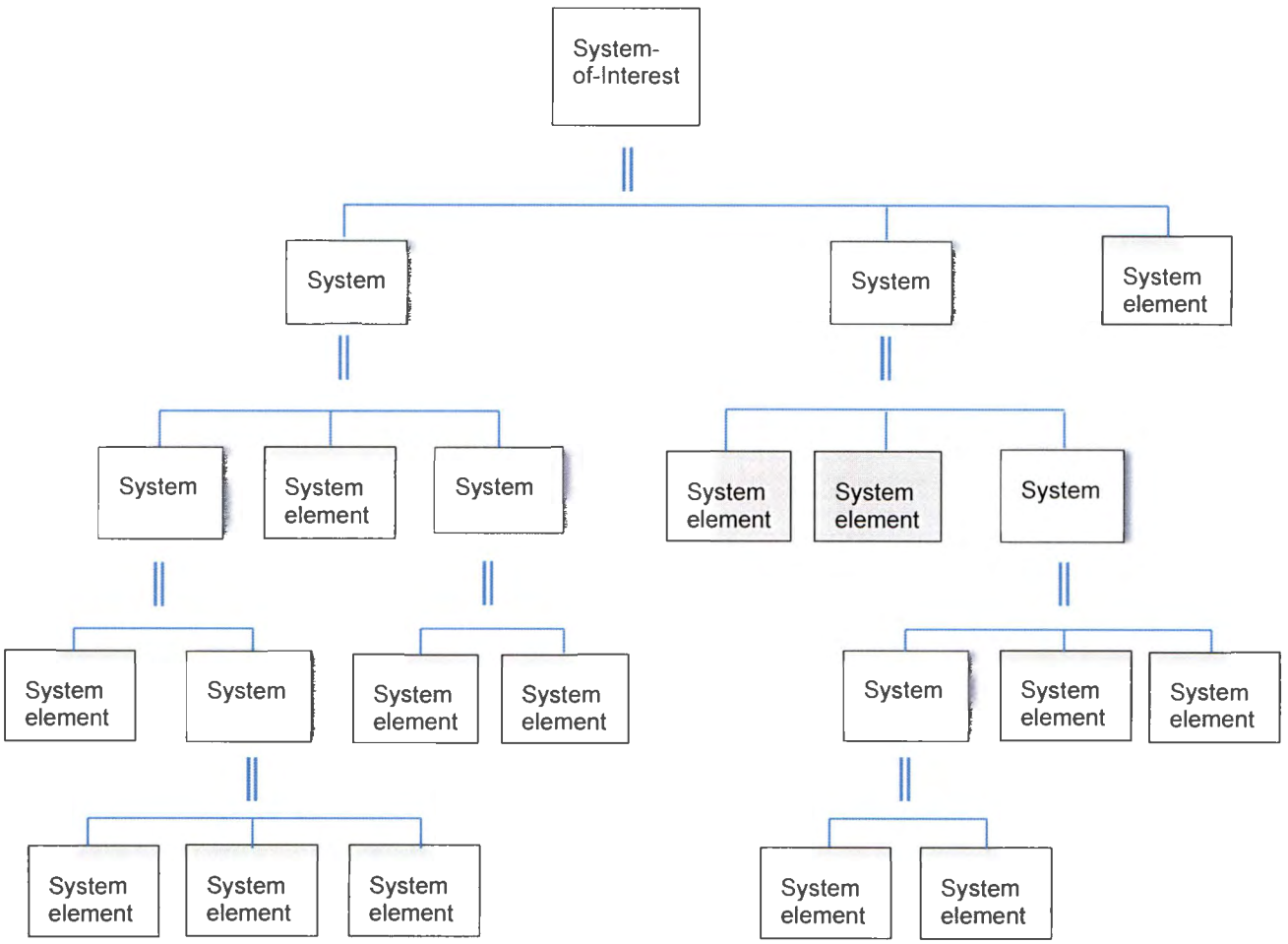

Figure 32: ISO/IEC 15288 System-of-interest structure

In this breakdown, the system-of-interest is decomposed into lower tier systems and system elements. The standard states that the perception and definition of a particular system, its architecture and its system elements depend on an observer's interests and responsibilities. The standard identifies the following key points regarding the characteristics of the system-of-interest:

a) Defined boundaries encapsulate meaningful needs and practical solutions;

b) There is a hierarchical or other relationship between system elements;

c) An entity at any level in the system-of-interest can be viewed as a system;

d) A system comprises an integrated, defined set of subordinate system elements;

e) Characteristic properties at a system's boundary arise from the interactions among system elements;

f) Humans can be viewed as both users external to a system and as system elements (e.g. operators) within a system;

g) A system can be viewed in isolation as an entity, i.e., a product, or as a collection of functions capable of interacting with its surrounding environment, i.e. a set of services.

The standard also identifies the concept of enabling systems, which are systems that do not form a direct part of the operational environment, but which facilitate the progression of the system-of-interest through its lifecycle. The standard states that, during a stage in the system lifecycle, the relevant enabling systems and the system-of-interest are considered together. Since these enabling systems are independent, they can also be considered as systems in their own right, and the system lifecycle processes can be applied to the enabling systems as well.

ISO/IEC 15288 presents a classification of:

- Agreement Processes;
- Organisational Project-Enabling Processes;
- Project Processes;
- Technical Processes.

The Agreement Processes and Organisational Project-Enabling Processes, listed below, can also be classified as enterprise support functions.

Aspects of interest to this study are described in the Organisational Project-Enabling Processes (Sub-clause 6.2), in particular the Lifecycle Model Management Process (Sub-clause 6.2.1); the Infrastructure Management Process (Sub-clause 6.2.2); and the Quality Management Process (Sub-clause 6.2.5), as well as in the Project Processes (Sub-clause 6.3), where the Configuration Management Process (Sub-clause 6.3.5) is of importance. The other sub-clauses describe aspects that are of importance to the project processes, which are excluded from the scope of the process design.

According to ISO/IEC15288, the Infrastructure Management Process (Sub-clause 6.2.2) provides "the enabling infrastructure and services to projects to support organisation and project objectives throughout the lifecycle." The standard further states that "[the Infrastructure Management Process] defines, provides and maintains the facilities, tools and communications and information technology assets".

The standard describes the concept of lifecycle as "an abstract functional model that represents the conceptualisation of the need for the system, its realisation, utilisation, evolution and disposal".

ISO/IEC15288 builds upon this concept by stating that:

> "[Lifecycle stages] represent the major lifecycle periods associated with a system and they relate to the state of the system description or the system itself. The stages describe the major progress and achievement milestones of the system through its lifecycle. They give rise to the primary decision gates of the lifecycle. These decision gates are used by organisations to understand and manage the inherent uncertainties and risks associated with costs, schedule and functionality when creating or utilising a system. The stages thus provide organisations with a framework within which organisation management has high-level visibility and control of project and technical processes."

The term "lifecycle model" is then defined as a "... framework of processes and activities concerned with the lifecycle that may be organised into stages, which also acts as a common reference for communication and understanding".

According to the standard, the purpose of the Lifecycle Model Management Process (Sub-clause 6.2.1) "is to define, maintain, and ensure availability of policies, lifecycle processes, lifecycle models, and procedures for use by the organisation with respect to IEEE STD 15288-2008". Note that this process has objectives in common with the quality management process, as described in Chapter 4.5.

In terms of the Quality Management Process (Sub-clause 6.2.5), the standard references ISO 9001 [29], which is discussed in Chapter 4.5 of the dissertation.

The standard discusses Configuration Management (Sub-clause 6.3.3) under Project Processes (Sub-clause 6.3). It references ISO 1007 [30], which is discussed in Chapter 4.6.

The standard describes a set of Technical Processes (Sub-clause 6.4) under the following headings:

a) Stakeholder Requirements Process;

b) Requirements Analysis Process;

c) Architectural Design Process;

d) Implementation Process;

e) Integration Process;

f) Verification Process;

g) Transition Process;

h) Validation Process;

i) Operation Process;

j) Maintenance Process;

k) Disposal Process.

In terms of this study, the operation, maintenance and disposal processes are not considered. The technical processes of interest are shown schematically in Figure 33, and are summarised below, under the headings as presented in the standard. Processes described in this standard, which support the technical processes and which are of relevance to this study, are also shown in the figure.

Stakeholder Requirements Definition (6.4.1)

- Identify stakeholders and elicit requirements from the identified stakeholders;
- Define constraints on the system solution;
- Identify the contextual use of the system;
- Identify human-system interactions;
- Specify stakeholder requirements and functions that relate to critical qualities (e.g. safety);
- Analyse the elicited requirements;
- Resolve requirements problems;
- Review requirement statements with stakeholders;
- Establish requirement database.

Requirements Analysis (6.4.2)

- Define the functional boundary of the system;
- Define the system functions;
- Define implementation constraints;
- Define performance and quality requirements;
- Specify safety requirements;
- Analyse the integrity of the requirements (ensure that each requirement is unique, complete, unambiguous, consistent with other requirements and verifiable);
- Review analysed requirements with stakeholders;
- Maintain the system requirements along with the associated rationale, decisions and assumptions throughout the system lifecycle.
- Demonstrate traceability between the system requirements and the stakeholder requirements;

Architectural Design (6.4.3)

- Define a logical system architecture;
- Allocate requirements to the system architecture and generate derived requirements as needed for the allocations;
- Define and document internal and external interfaces;
- Analyse the architecture to establish design criteria for each element;
- Identify requirements that relate to human-system interaction;
- Identify off-the-shelf solutions for architectural elements;
- Perform architectural design trade-off studies;
- Specify a physical architectural design solution;
- Record the architectural design information.

Implementation (6.4.4)

- Generate an implementation strategy (decided on procedures, processes, tools and equipment to be used in the implementation process);
- Identify constraints on the design solution, imposed by the implementation strategy and implementation technology;
- Realise or adapt system elements – fabricate hardware and create software;
- Record evidence that realised system elements comply with its specifications;
- Package and store system elements.

Integration (6.4.5)

- Define an assembly sequence and strategy that minimises system integration time, costs, and risks;
- Identify constraints on the design arising from the integration strategy;
- Obtain integration enabling systems and specified materials;
- Obtain system elements;
- Assure system elements have been verified and validated against its specified acceptance criteria;
- Integrate system elements;
- Analyse, record and report integration information.

Verification (6.4.6)

- Define a verification strategy;
- Define a verification plan based on system requirements;

- Ensure the verification environment is available and other preparations for verification are complete;
- Conduct verification to demonstrate compliance to specified design requirements;
- Make verification data available;
- Analyse, record and report verification, discrepancy and corrective action information.

Transition (6.4.7)

- Prepare a transition strategy;
- Prepare the site of operation in accordance with installation requirements;
- Deliver the system for installation at the correct location and time;
- Install the system and interface it to its environment;
- Demonstrate the proper installation of the system;
- Activate the system;
- Demonstrate that the installed system is capable of delivering its required services;
- Demonstrate that the services provided by the system are sustainable by the enabling systems;
- Analyse, record and report transition information.

Validation (6.4.8)

- Define a strategy for validating the system in its target environment;
- Prepare a validation plan;
- Ensure that operators, enabling systems and required facilities are ready for validation;
- Conduct system validation to demonstrate conformance with stakeholder requirements;
- Analyse, record and report transition information.

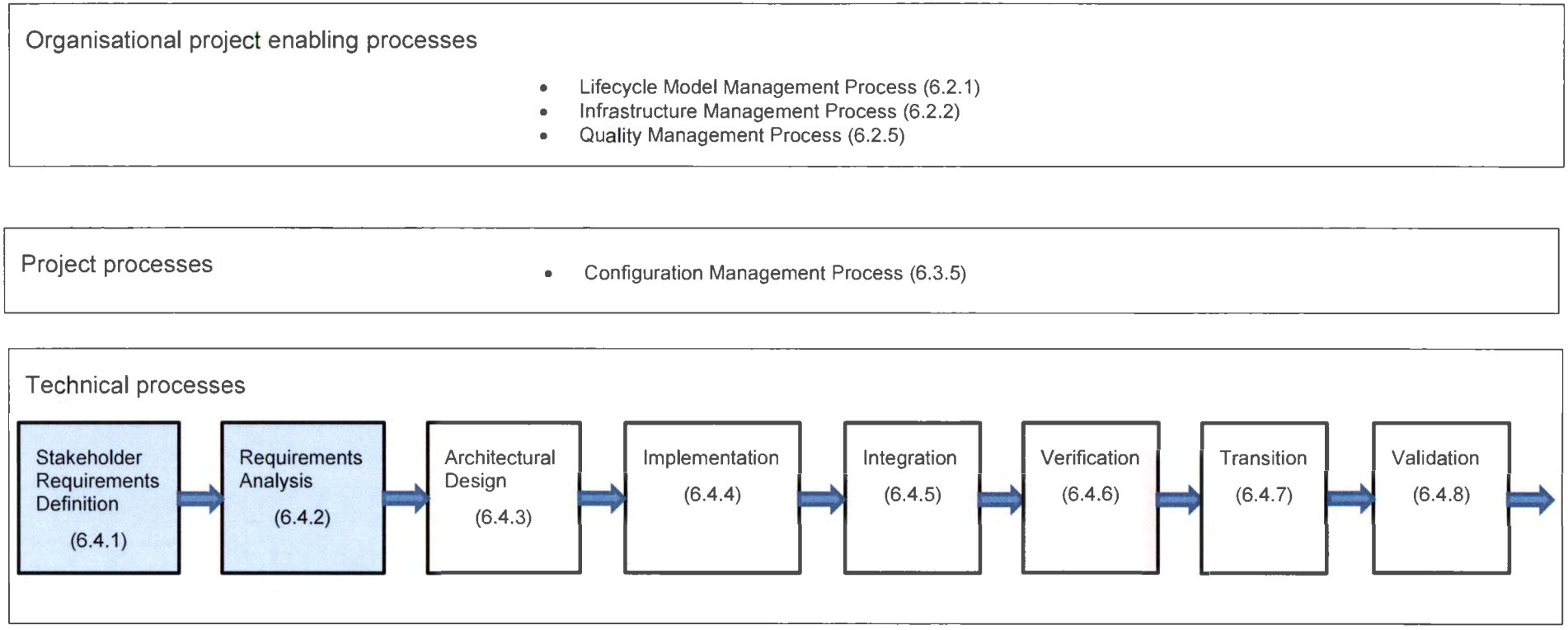

Figure 33: ISO/IEC 15288 lifecycle model (technical processes up to completion of development)

4.4.4 Synopsis of ISO/IEC 15288

4.4.4.1 Scope of engineering activities

The scope of the technical processes is distinctly defined in this standard in terms of the activities and tasks that are required to define and realise a system-of-interest. This outlining is also not in conflict with the definition of technical processes as defined by ANSI/EIA-632.

The identification of technical processes as indicated by this standard will therefore be used in the process design presented in Chapter 5.

The standard highlights the fact that the lifecycle processes should be applied recursively to the system-of-interest and its system elements, as well as to the development of the enabling systems.

This aspect of recursive use of the lifecycle processes on all layers of system hierarchy, as well, as on the level of enabling systems, is very important in terms of the identification of the process scope with reference to this study.

4.4.4.2 Definition of activities and tasks

The standard clearly identifies activities and tasks associated with each of the technical processes as described in Sub-clause 6.4. However, inspection of the activities, as they are identified for each of the technical processes, reveals that some activities can be considered as "core" activities, i.e. activities that directly produce lifecycle data and elements related to the definition of the system, and activities which are concerned with the verification and validation of outputs of the "core" activities. It will be shown in Chapter 4.7 that in some instances it is required to perform these tasks with independence, therefore the description of these verification and validation tasks need to be isolated from the "core" tasks and activities in the process design. Also note that the verification process, as described in Sub-clause 6.4.6, is different from these activities, as this process verifies that the system-of-interest complies with its requirements, whereas the "embedded" verification and validation activities are concerned with the correctness and completeness of all individual process outputs.

This distinction between the verification process described by the standards, and the embedded verification and validation activities is very important in terms of the derivation of the lifecycle model on which the process design is based.

The Transition Process (6.4.7) is normally not applicable for platform specific airborne electronic systems, as the verification process requires that the system is operated in its target environment, i.e. the aircraft type for which it is intended, and that the system can be considered to be deployed after completion of the verification process.

For the type of systems considered in this study, the Validation Process (6.4.8) will typically serve to validate the User Requirements, which was an input to the requirements process, and is also not considered in the process design.

4.4.4.3 Development process framework

This standard clearly defines the concept of a lifecycle model.

It must be noted that similarities exist between the lifecycle model represented in Figure 33 and the lifecycle model identified by ANSI/EIA-632, as shown in Figure 31. The objective of the requirements processes (6.4.1 and 6.4.2) is to establish a set of validated requirements, as also indicated in ANSI/EIA-632. The architecture design process (6.4.3) is similar to the solution definition process of ANSI/EIA-632, as indicated in Figure 31, and aims at identifying the major elements of the system-of-interest and their interfaces. The implementation (6.4.4) and integration processes (6.4.5) realise a system-of-interest which is subjected to the verification process (6.4.6). The verified system-of-interest is deployed in its intended environment by means of the transition process (6.4.7) and it is confirmed that the system fits its intended purpose through the validation process (6.4.8).

The similarities between the lifecycle models presented by ISO/IEC 15288 and ANSI/EIA-632 indicate that there are elements of the lifecycle models which are generic to development processes in general. This important aspect will be explored further in this research.

4.4.4.4 Development process controls

The following lifecycle processes described in the standard can be construed as technical control processes: The Quality Management Process (6.2.5), the Configuration Management Process (6.3.5) and the Measurement Process (6.3.7). The measurement process described in the standard is generic, and the cited activities, applicable to technical processes, are described under quality management processes in e.g. SAE AS9100 [27], which is discussed in Chapter 4.5.2.

4.4.5 IEEE-1220

- Definition of a system and process

International Standard ISO/IEC 26702 (IEEE STD 1220-2005): *Systems engineering – Application and management of the systems engineering process* [38] presents an approach for the definition and management of systems. Clause 4 of the standard identifies general requirements for the planning and implementing of a systems engineering capability within an enterprise.

The standard defines a system as "... a set or arrangement of elements and processes that are related and whose behaviour satisfies customer/operational needs and provides for lifecycle sustainment of the products". Importantly, the standard takes the view that the lifecycle processes can also be viewed as a system where products are developed to meet with the objectives of the process, and that these products may in turn require lifecycle sustainment processes and products.

- Policies and procedures for systems engineering

The standard requires that the enterprise develops and maintains policies and procedures for systems engineering, and it indicates that policies and procedures provide for:

a) Application of the systems engineering process throughout a project lifecycle;
b) Preparation and approval of a systems engineering management plan;
c) Preparation of a systems engineering master schedule and systems engineering detailed schedule;
d) Preparation and approval of the integrated data package;
e) Monitoring and reporting technical progress of the project;
f) Preparation for, and the conduct of, technical reviews;
g) Contents and maintenance of an integrated repository;
h) Continuous improvement of products and processes.

- Systems engineering and lifecycle processes

IEEE-1220 describes the Systems Engineering Process (SEP) as:

> "[A] generic problem-solving process that provides the mechanisms for identifying and evolving the product and process definitions of a system. The SEP applies throughout the system lifecycle to all activities associated with product development, verification/test, manufacturing, training, operation, support, distribution, disposal, and human systems engineering."

The standard provides a detailed description of this process, which is summarised below.

- The enterprise should perform comprehensive planning of the project and apply the SEP at all levels;
- The progress of the project should be controlled using technical reviews at each level of development and risk, data, interface, configuration, and performance based progress management should be applied;
- Models and prototypes should be generated to support trade-off decisions;
- An integrated data package should be generated containing the design details of the system and the outputs of all technical activities should be captured in an integrated repository.

- Lifecycle stages

The system lifecycle is defined as "the system or product evolution, initiated by a perceived stakeholder need through the disposal of the products". The lifecycle processes and aspects with which they deal are identified by this standard as:

- Development and test processes: equipment, procedures, software applications, computer resources, personnel and suppliers/vendors;
- The manufacturing process: equipment/tools, procedures, software applications, computer resources, parts inventory, personnel, suppliers/vendors and quality control;
- The distribution and support processes: facilities, equipment/tools, procedures, software applications, computer resources, spare parts inventory, maintainers and suppliers/vendors;
- Operations and training processes: facilities, equipment/tools, procedures, software applications, computer resources, operators and trainers;
- A disposal process: facilities, equipment and tools, procedures, and personnel.

Clause 5 of the standard considers the application of systems engineering throughout the system lifecycle. The standard describes the following stages of development and operations:

Stages of development:

a) System definition;

b) Subsystem definition;

1) Preliminary design of subsystems;

2) Detailed design of subsystem components;

3) Fabrication, assembly, integration and test.

Stages of operations:

a) Production;

b) Support.

As explained above, for the purpose of this study only the "stages of development" are addressed here.

A schematic representation of the lifecycle described by IEEE 1220, for the stages of development, is presented in Figure 34.

During the system definition stage the system concept is defined and the subsystems and subsystem interfaces (i.e. the system architecture) are identified, along with the identification of the human/system interfaces and the identification of system quality factors. A set of system level specifications and other preliminary lower level specifications are developed and placed under configuration management. The engineering plans for the project are developed during the system definition stage. An alternate concept review is conducted at an appropriate juncture in the system definition stage, if needed. The stage is concluded by a system definition review, and a system baseline and subsystem design-to baselines are established.

The system definition stage is followed by the preliminary design stage. The objectives of this stage are to:

- Identify the assemblies and components (together with their interfaces that will constitute the system);
- Mitigate subsystem and component risks;
- Identify the HMI aspects and
- Identify and quantify lifecycle quality factors at the level of assemblies and components.

Subsystem level specifications and preliminary component specifications are developed, specifications developed during the system definition stage are updated, and these are all placed under configuration management. The engineering plans are updated as necessary to guide future activities. Subsystem reviews are conducted at appropriate points in the stage as required. The preliminary design stage is concluded

by a system definition review, and updated system design-to baselines, as well as component build-to baselines, are established.

After the completion of the preliminary design stage, the detailed design stage is initiated. During this stage, component definitions are developed, component level risks are mitigated, and HMI aspects and lifecycle quality factors are identified at the components level. Specifications are updated or developed as required, and an integrated data package containing detailed drawings, code listings, etc. is produced. The engineering plans are updated as required to guide the fabrication, assembly, integration, and test activities that follow this stage. Component and subsystem reviews are conducted as required. The detailed design stage is concluded by a system definition review, and updated system and design-to baselines, as well as updated build-to baselines, are established.

The development process is concluded by the fabrication, assembly, integration, and test (FAIT) stage. Hardware components are fabricated and software components are implemented. Components, assemblies and subsystems are assembled and integrated up to the system level. The standard indicates that the purpose of the FAIT stage is to verify that the products designed in the course of the preceding stages satisfy specifications. Test readiness reviews are conducted to ensure the correctness of test procedures, to ensure that the test environment is satisfactory for the tests and that the test objects are at a sufficient level of maturity to be subjected to tests. Verification testing is commenced at the component level, and these are integrated into assemblies and subjected to verification tests, continuing this sequence up to the system level. Failures and deficiencies detected during the testing process are analysed and fixed, and retested. Functional configuration audits are conducted at the level of components, assemblies, subsystems and at the system level to verify that products have achieved requirements. Production approval reviews are conducted to confirm that the system can transition to the stages of operation.

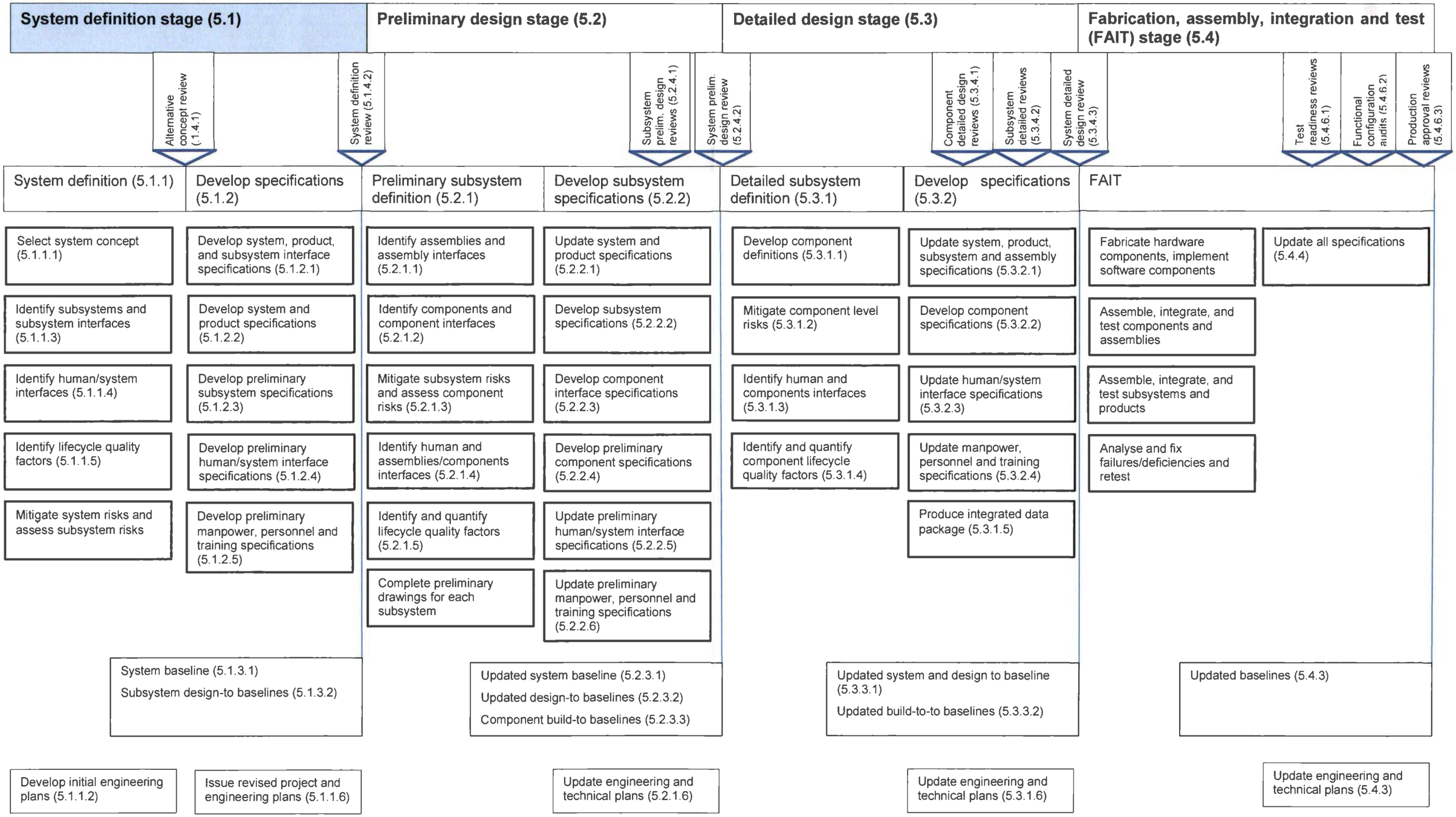

Figure 34: IEEE-1220 lifecycle model – stages of development

- Systems engineering process

The standard describes the systems engineering process separately from the lifecycle stages. The systems engineering process is presented schematically in Figure 35, which is an interpretation of Figure 4, Clause 4, of the standard.

The systems engineering process is described in a high level of detail in the standard, and only salient points are summarised here.

The standard distinguishes between precedented systems, for which design examples exists within its class, and unprecedented systems, for which this is not the case. For precedented systems, guidance with respect to system architecture and specifications exists to some extent, whereas no examples exist that can be used during the development of unprecedented systems. This differentiation is important as there is a significant difference in the approach to develop a precedented system than an unprecedented system due to the different levels of uncertainties.

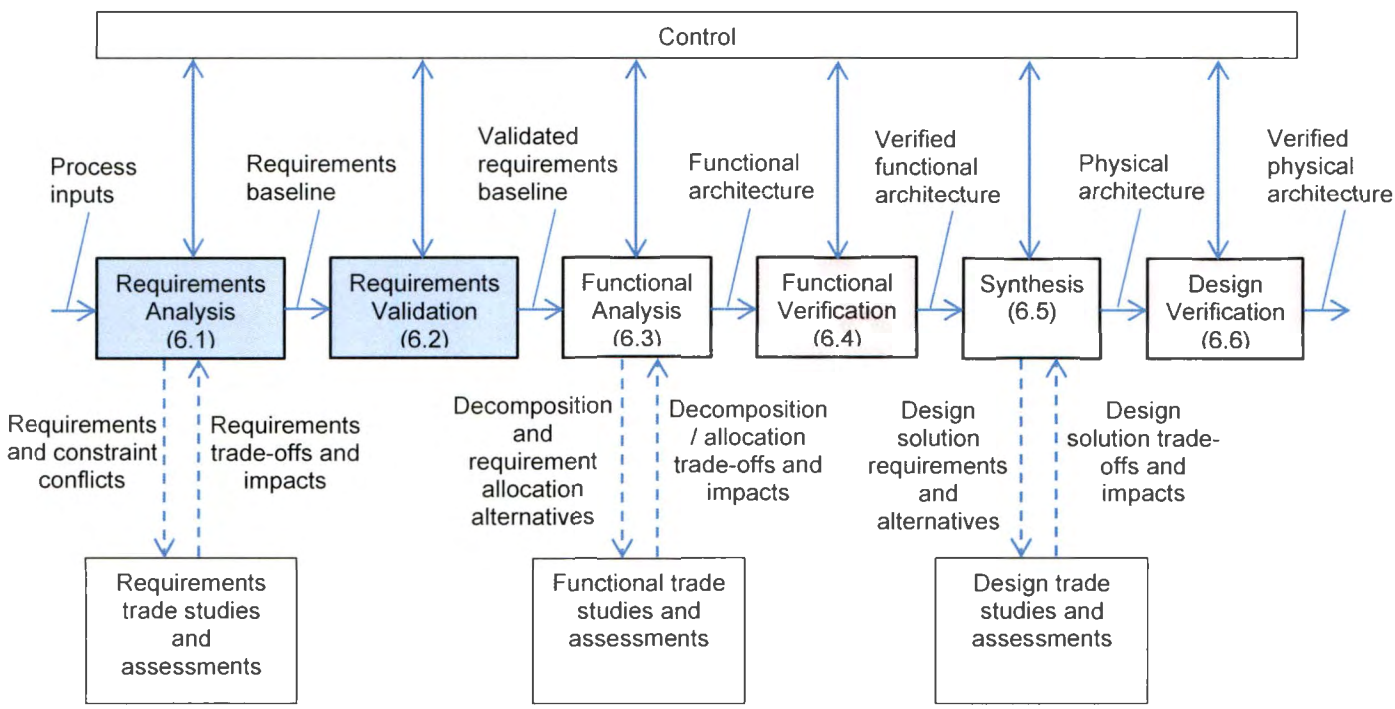

Figure 35: Systems engineering process (IEEE-1220)

The standard provides detailed guidance on each of the sub-processes depicted in this diagram.

4.4.6 Synopsis of IEEE-1220

4.4.6.1 Scope of engineering activities

The scope of the development process, as presented by IEEE-1220, can be inferred from the list of policies and procedures it specifies. Referring to the distinction between technical process and project process as presented by ANSI/EIA-632, an inspection of this list shows that IEEE-1220 does not segregate activities that are concerned with achieving the development progress schedule from the activities that affect the definition and attributes of the system.

The scope of the process, as implicitly identified in IEEE-1220, does not contradict the segregation as delineated by ANSI/EIA-632, but it is not as clearly defined. Therefore, IEEE-1220 will not be used to support the determination of the scope of engineering activities.

4.4.6.2 Definition of activities and tasks

This standard describes the systems engineering process to a high level of detail, in a firm top-down approach. However, the process details may prove cumbersome to implement directly as it is presented in the standard, as all the tasks identified may not be applicable to the development of particular products. The information contained in this standard is, as in the instance of MIL-STD-498, useful as a checklist in the design of a development process.

Note that this standard also promotes models and prototypes to support trade-off decisions, i.e. development process enablers.

4.4.6.3 Development process framework

This total lifecycle is referred to in EIA-632 as the enterprise based lifecycle, i.e. the lifecycle of the system from conception throughout its utilisation up to its final disposal. As explained in the previous section, this lifecycle is not considered in this study. However, the topics as they are listed above serve as a guide when planning the development of enabling systems.

Conceptually, there are no significant differences between the stages up to this point and the requirements management and solution definition stages from ANSI/EIA-632 [12] and shown in Figure 31, and information from these stages will be considered in the process design.

The FAIT stage is viewed as a single lifecycle stage in IEEE-1220, which is different to the views portrayed by other literature considered in this study. In the view presented by this standard, the system development follows from system to subsystem to component definition and then realisation, integration and testing in a bottom-up view, rather than considering the system-of-systems view. This implies a single-step integration approach, which was shown to be undesirable in Chapter 4.3.4. The use of test and integration loops, shown in Section 4.3.4 and Figure 24, is preferable for the type of system described in this study.

4.4.6.4 Development process controls

It was pointed out in section 4.4.1 that the development process shall be assessed against the project plans. IEEE-1220, in point of fact, states that development should be executed against policies and procedures maintained by the enterprise.

The role of policies and procedures as managed by the enterprise, in controlling the development process, need to be aligned with the project agreement.

4.4.7 ISO/IEC 12207

ISO/IEC 12207:2008 *Information technology – Software lifecycle processes* [47] identifies three classes of lifecycle processes, i.e. primary lifecycle processes, supporting lifecycle processes and organisational lifecycle processes.

The following five primary (software) lifecycle processes are identified:

1. Acquisition process (Sub-clause 5.1): defines the activities of the acquirer (of the software product or services);
2. Supply process (Sub-clause 5.2): defines the activities of the supplier (of the software product or services);
3. Development process (Sub-clause 5.3): defines the activities of the developer of the software product;
4. Operation process (Sub-clause 5.4): defines the activities of the operator;
5. Maintenance process (Sub-clause 5.5): defines the activities of the organisation that maintains the software product, i.e. managing modifications to the software product to keep it current and in operational fitness.

The development process described in Sub-clause 5.3 is of particular interest to this study. The standard identifies thirteen activities which comprise this process, and these are summarised below.

The process implementation activity (Sub-clause 5.3.1) firstly entails the definition or selection of a software lifecycle model appropriate to the scope, magnitude and complexity of the project, to which the activities and tasks of the development process can be mapped. The standard explicitly mentions that these activities and tasks may overlap or interact or may be performed iteratively or recursively. Standards, methods, tools, and programming languages for performing the activities of the development process are selected and tailored where required and plans are developed for conducting the development process.

The system requirements analysis activity (Sub-clause 5.3.2) develops system requirements. The functions and capabilities of the intended system are identified and documented. The system requirements are evaluated according to the following criteria:

a) Traceability to acquisition needs;

b) Consistency with acquisition needs;

c) Testability;

d) Feasibility of commencing system architectural design;

e) Feasibility of operation and maintenance.

A top-level architectural design of the system is established during the system architectural design activity (subclause 5.3.3). Hardware and software items and manual operations are identified, and system requirements are allocated to the identified items. These items are then detailed in terms of hardware and software configuration items. The system architecture is evaluated according to the criteria listed below.

a) Traceability to the system requirements;

b) Consistency with the system requirements;

c) Appropriateness of design standards and methods used;

d) Feasibility of the software items fulfilling their allocated requirements;

e) Feasibility of operations and maintenance.

During the software requirements analysis activity (subclause 5.3.4) the software requirements for each software configuration item shall be established and documented. The software requirements are evaluated according to the following criteria:

a) Traceability to system requirements and system design;

b) External consistency with system requirements;

c) Internal consistency;

d) Testability;

e) Feasibility of commencing software design;

f) Feasibility of operation and maintenance.

This evaluation is to be performed at a joint review between the acquirer and the supplier, and a requirements baseline shall be established upon successful completion of the review.

The software architectural design activity (subclause 5.3.5) transforms the software requirements into an architecture that describes its top-level structure and identifies the software components. It is ensured that the requirements for the software item are allocated to system components and the software architecture is documented. A top level design for the interfaces external to the software item, and between the software components of the item, is developed. Preliminary versions of user documentation and test requirements are developed, and the schedule for software integration is determined. The architecture of the software item is evaluated, considering the following criteria:

a) Traceability to software requirements;

b) External consistency with software requirements;

c) Internal consistency between software components;

d) Appropriateness of design methods and standards used;

e) Feasibility of commencing with detailed design;

f) Feasibility of operation and maintenance.

This evaluation is to be performed at a joint review between the acquirer and the supplier.

A detailed design for each software component of the software item is developed and documented by the software detailed design activity (subclause 5.3.6). Software components are refined into lower level units that can be coded, compiled and tested. The design includes detailed design of external and internal interfaces. The detail of the design should be such that it permits coding requiring no further information. The user documentation shall be updated as necessary. Test requirements and the

schedule for testing software units are to be defined and documented. Test requirements should include stressing the software unit at the limit of its requirements. The software detailed design and the test requirements shall be evaluated against the following criteria:

a) Traceability to software requirements;

b) External consistency with architectural design;

c) Internal consistency between software components and software units;

d) Appropriateness of design methods and standards used;

e) Feasibility of testing;

f) Feasibility of operation and maintenance.

The software units, as well as test procedures and data for testing the software units are developed during the software coding and testing activity (subclause 5.3.7). Software code and unit test results are evaluated and documented, using the following criteria:

a) Traceability to software requirements and the design of the software item;

b) External consistency with the requirements and the design of the software item;

c) Internal consistency between unit requirements;

d) Test coverage of units;

e) Appropriateness of coding methods and standards used;

f) Feasibility of commencing software integration and testing;

g) Feasibility of operation and maintenance.

The software units and components constituting the software configuration item are integrated during the software integration activity (subclause 5.3.8). The integration process shall be guided by a documented integration plan which contains test requirements, procedures, data, responsibilities and schedule. For each qualification requirement of the software item, a set of tests, (test cases (inputs, outputs, test criteria), and test procedures for conducting software qualification testing, is to be developed and documented. The integration plan is evaluated against the following criteria:

a) Traceability to the system requirements;

b) External consistency with the system requirements;

c) Internal consistency;

d) Test coverage of the requirements of the software item;

e) Appropriateness of test methods and standards used;

f) Conformance to expected results;

g) Feasibility of commencing software qualification testing;

h) Feasibility of operation and maintenance.

This evaluation is to be performed at a joint review between the acquirer and the supplier.

The implementation for each software configuration item is tested for compliance during the software qualification testing activity (subclause 5.3.9).

The standard provides the following definitions regarding qualification:

> Qualification is the process of demonstrating whether an entity is capable of fulfilling specified requirements;
>
> A qualification requirement is a set of criteria or conditions that have to be met in order to qualify a software product as complying with its specifications and being ready for use in its target environment;
>
> Qualification testing comprises testing, conducted by the developer and witnessed by the acquirer, to demonstrate that a software product meets its specifications and is ready for use in its target environment.

After completion of the software qualification tests, the design, code, tests, test results and user documentation is evaluated against the following criteria:

a) Test coverage of the requirements of the software item;

b) Conformance to expected results;

c) Feasibility of commencing system integration and testing;

d) Feasibility of operation and maintenance.

After completion of the tasks that constitute this activity, an audit (described in subclause 6.7) shall be conducted. A software qualification baseline shall be established.

During the system integration activity (subclause 5.3.10), the software configuration items, hardware configuration items, manual operations, and other systems as necessary, are integrated into the system and tested against their requirements. The

standard indicates that the integration and test results shall be documented. As part of the integration activity, a set of tests, test cases, and test procedures is to be developed for each qualification requirement of the system. The developer shall ensure that the integrated system is ready for qualification testing. The integrated system is evaluated against the following criteria:

a) Test coverage of system requirements;

b) Appropriateness of test methods and standards used;

c) Conformance to expected results;

d) Feasibility of commencing system qualification testing;

e) Feasibility of operation and maintenance.

The system qualification testing activity (subclause 5.3.11) ensures that the implementation of each system requirement is tested for compliance and that the system is ready for delivery. After completion of qualification tests, the system is evaluated against the following criteria:

a) Test coverage of system requirements;

b) Conformance to expected results;

c) Feasibility of operation and maintenance.

After completion of the system qualification activity, an audit (described in subclause 6.7) shall be conducted. A system qualification baseline shall be established.

The software installation activity (subclause 5.3.12) installs the system in the target environment. For airborne systems this can e.g. entail the installation of the system in the aircraft. The standard states that this shall be according to an installation plan.

The final step in the development process is the software acceptance support activity (subclause 5.3.13). This activity supports the acceptance reviews and tests of the software product conducted by the acquirer. This acceptance review and testing considers the results of the joint reviews, audits, software and system qualification testing.

The development process described in subclause 5.3 of the standard is shown schematically in Figure 36.

The standard describes eight supporting lifecycle processes. These are summarised below; aspects relevant to this study are described in slightly more detail.

1 The Documentation process (subclause 6.1) defines the activities for recording the information produced by a lifecycle process or activity. This process includes the planning of all the documents to be produced during the lifecycle of a software product.

Prepared documents should be reviewed to confirm correctness of format, technical content, and presentation style against their documentation standards, and documents shall be approved by authorised personnel prior to issue.

Documents should be controlled in accordance with the Configuration Management Process.

2 The Configuration management process (subclause 6.2) defines the configuration management activities. This process applies "administrative and technical procedures throughout the software lifecycle to: identify, define, and baseline software items in a system; control modifications and releases of the items; record and report the status of the items and modification requests; ensure the completeness, consistency and correctness of the items; and control storage, handling, and delivery of the items.

The description of these activities corresponds with the processes described in Chapter 4.6.

3 The Quality assurance process (subclause 6.3) defines the activities for objectively assuring that the software products and processes are in conformance with their specified requirements and adhere to their established plans. Importantly, it is indicated that quality assurance may make use of the results of other supporting processes, such as Verification, Validation, Joint Reviews, Audits and Problem Resolution. The standard identifies three aspect of quality assurance, i.e. Product assurance; Process assurance; and Assurance of quality systems.

3.1 The Product assurance activity (sub-clause 6.3.2) provides assurance that plans required by the contract are documented and in compliance with the contract; and that software products and related documentation comply with the contract and in adherence to the relevant plans.

3.2 The Process assurance activity (sub-clause 6.3.3) provides assurance that the software lifecycle processes employed for the project comply with the contract and adhere to the plans. It also provides assurance that internal software engineering practises, development environment, test environment, and libraries

comply with the contract. It further also provides assurance that the software product and process measurements are in accordance with established standards and procedures, and that the staff assigned to the project have the skill and knowledge appropriate for the tasks.

3.3 Assurance of quality management systems (sub-clause 6.3.4) shall be in accordance with ISO 9001, which is described in Chapter 4.5 of this dissertation.

4 The Verification process (subclause 6.4) defines the activities (for the acquirer, the supplier, or an independent party) for verifying the software products in varying depth depending on the software product.

The standard addresses the concept of independence. This aspect is addressed in more detail in Chapter 4.7.4, and thus not elaborated here.

The standard identifies seven types of verification. These are listed below, and aspects related to technical aspects of the development process are summarised. Verification related to project and organisational matters, as described in these sub-clauses, are not considered here.

4.1 Contract verification (sub-clause 6.4.2.10):

The contract shall be verified to confirm that the requirements are consistent and that they cover user needs; and that acceptance criteria and procedures are stipulated in accordance with the requirements. It shall also be verified that adequate procedures for handling changes to requirements and escalating problems are stipulated.

4.2 Process verification (sub-clause 6.4.2.2):

It shall be verified that the processes selected for the project are adequate, implemented, being executed as planned, and compliant with the contract, and that the standards, procedures, and environments for the project's processes are adequate.

4.3 Requirements verification (sub-clause 6.4.2.3):

The system requirements shall be verified to ensure they are consistent, feasible, and testable, and that they have been appropriately allocated to hardware items, software items, and manual operations.

The software requirements shall be verified to ensure they are consistent, feasible, and testable, and that they accurately reflect system requirements. It

shall be verified that software requirements related to safety, security, and criticality are correct.

4.4 Design verification (sub-clause 6.4.2.3)

The design shall be verified to confirm that the design is correct and consistent with and traceable to requirements. In terms of software design, it must be ensured that the design implements proper sequence of events, inputs, outputs, interfaces, logic flow, allocation of timing and sizing budgets, and error definition, isolation and recovery. It must be shown that the design implements safety, security, and other critical requirements correctly.

4.5 Code verification (sub-clause 6.4.2.3)

It must be verified that the code is traceable to design and requirements, testable, correct, and compliant with requirements and coding standards. It must also be shown that the code implements appropriate software measures to ensure correct and error free operation.

4.6 Integration verification (sub-clause 6.4.2.3)

It shall be verified that all the hardware, software, and manual operation elements of the system have been integrated completely and correctly.

4.7 Documentation verification (sub-clause 6.4.2.3)

It must be verified that documentation is accurate, complete and consistent, and that configuration management of documents followed specific procedures.

5 The Validation process (subclause 6.5) defines the activities (for the acquirer, the supplier, or an independent party) for validating the software products of the software project. It must be determined whether the requirements, and the final, as built system or software product fulfils its intended use.

6 The Joint review process (subclause 6.6) defines the activities for evaluating the status and products of an activity ... in a joint forum.

The standard makes a clear distinction between technical and project management reviews. Technical reviews are described in sub-clause 6.7.3. These reviews shall evaluate software products or services and provide evidence that the products or services under consideration are complete, that they comply with their standards and specifications, changes are orderly implemented and that work is performed in accordance with the plans, standards and guidelines of the project.

7 The Audit process (subclause 6.7) defines the activities for determining compliance with the requirements, plans and contract. The standard indicates that auditing personnel may not have any direct responsibility for the software products and activities they audit.

The audits should confirm the following:

a) The software products reflect the design documentation;

b) The acceptance review and test requirements prescribed by the documentation are adequate for the acceptance of the software products;

c) Test data comply with the specification;

d) Software products were successfully tested and meet with their specifications;

e) Test reports are correct and discrepancies between actual and expected results have been resolved;

f) User documentation complies with the standards as specified;

g) Activities have been conducted according to applicable requirements, plans, and contract;

8 The Problem resolution process (subclause 6.8) defines a process for analysing and removing the problems (including non-conformances), whatever their nature or source, that are discovered during the execution of development, operation, maintenance, or other purposes.

Four organisational lifecycle processes are identified. These are:

1. Management process (subclause 7.1): defines the basic activities of the management, including project management, during a lifecycle process;
2. Infrastructure process (subclause 7.2): defines the basic activities for establishing the underlying structure of a lifecycle process;
3. Improvement process (subclause 7.3): defines the basic activities that an organisation performs for establishing, measuring, controlling, and improving its lifecycle process;
4. Training process (subclause 7.4): defines the activities for providing adequately trained personnel.

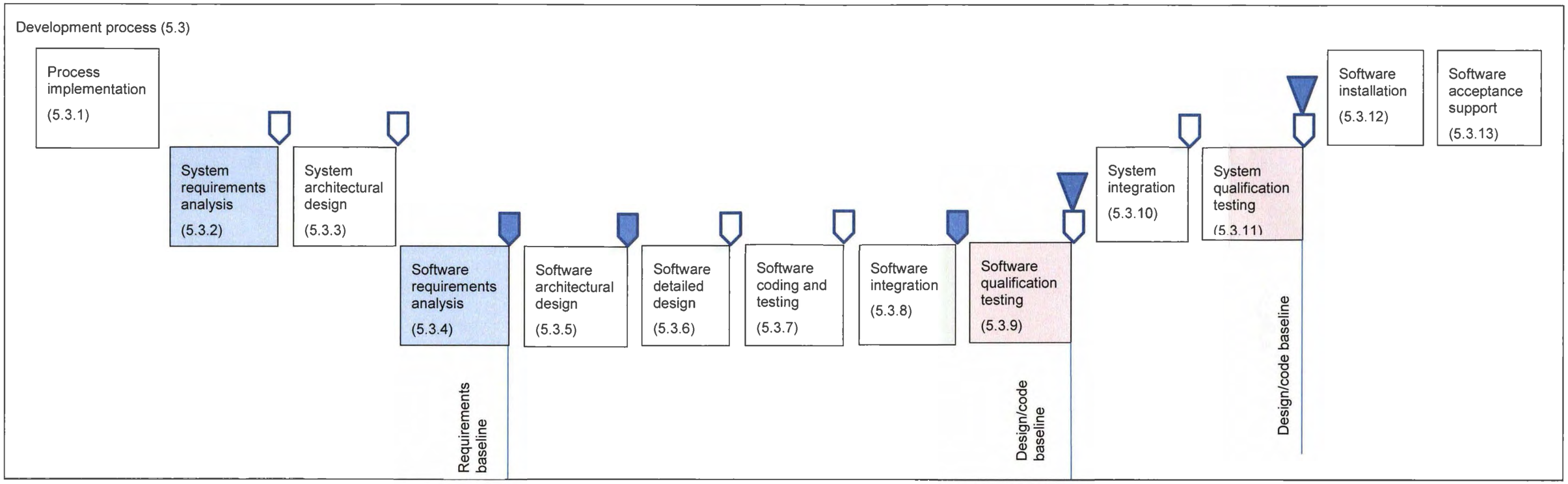

Figure 36: ISO/IEC 12207 software development process

4.4.8 Synopsis of ISO/IEC 12207

4.4.8.1 Scope of engineering activities

The standard presents a well-defined scope for the development process, in sub-clause 5.3, and are similar to the technical processes described by ISO/IEC15288:2008(E) [39]. The activities described in sub-clause 5.3 of the standard apply to the development of the system employing the software, as well as the development of the software.

The identification of the development process elements as described in sub-clause 5.3 of the standard will be used to augment the identification of technical processes indicated by ISO/IEC15288:2008(E) [39].

4.4.8.2 Definition of activities and tasks

Although primarily focussing on software development, the tasks and activities described by this standard are similar to those described by the other contemporary standards reviewed in this chapter.

The notions of qualification, verification, and validation are treated differently by most of the references consulted in this study. The concepts of qualification, verification, evaluation and validation, as discussed in ISO/IEC 12207, provide a basis against which these concepts can be examined.

ISO/IEC 12207 describes qualification testing, (sub-clauses 5.3.9 and 5.3.11), which follows software and system integration, as tests which are executed to demonstrate that a software product meets its specifications and is ready for use in its target environment. ISO/IEC15288:2008(E) [39] describes a verification process (sub-clause 6.4.6) which is also performed after completion of the integration process, which demonstrates compliance to specified design requirements. As the verification tasks implied by ISO/IEC15288:2008(E) are not limited to tests, i.e. it can include analyses and reviews, the verification process described by this standard encompasses a wider scope than the qualification tests described by ISO/IEC 12207. However, it is evident that the intention of this technical process is the same as the qualification tests described by ISO/IEC 12207, i.e. to provide evidence of compliance to the system and software requirements.

The use of verification as a development control mechanism, as interpreted by ISO/IEC 12207 is described in section 4.9.4.

4.4.8.3 Development process framework

The lifecycle model implied by ISO/IEC 12207, as shown schematically in Figure 36. Inspection of this diagram shows significant similarity to the lifecycle model described by MIL-STD-498 [10] and shown in Figure 17.

4.4.8.4 Development process controls

A number of the supporting lifecycle processes described in clause 6 of ISO/IEC 12207 can be considered to be processes which control aspects of the development process.

As a control mechanism, the Configuration Management process identifies specific software items in a system and control modifications and releases of these items. The process also ensures the completeness, consistency and correctness of the items.

The Quality Assurance process controls the integrity of the software items as it provides assurance that the software products and processes are in conformance with their requirements and adhere to the plans where the development processes are defined. It also provides assurance that internal software engineering practises, development environment, test environment, and libraries comply with the contract. It further provides assurance that the software product and process measurements are in accordance with established standards and procedures.

The Verification process as described by this standard confirms correctness, consistency, compliancy to standards, traceability, and testability of the requirements, design, implementation, and integration of the software and system. This process also confirms that documentation is accurate, complete and consistent, and that configuration management of documents is appropriate.

The Audit process provides evidence that the products or services under consideration are complete, that they comply with their standards and specifications, changes are orderly implemented and that work is performed in accordance with the plans, standards and guidelines of the project.

Although these control mechanisms are defined in detail, scrutiny of the details show significant overlaps. It follows that these overlaps can be rationalised significantly by judicious design of the control mechanisms in a detailed process definition.

4.4.9 Process constructs

A number of detailed process descriptions were described in the previous sections, as they were represented in widely used standards. These process descriptions were

assessed in terms of the research requirements that were formulated in Chapter 3. In this section, a methodology for the representation of a process in terms of an elemental and consistent construct is examined, in order to augment the abstraction and generalisation of the concepts underlying the development process. Although this topic does not relate to the research requirements, this methodology will be used to further analyse process descriptions in the following sections, and to rationalise the design of the process, described in Chapter 5.

ISO/IEC15288:2008(E) [39], Annex C, provides informative (as opposed to normative) guidance regarding process integration and process constructs.

The grouping of the processes according to ISO/IEC15288:2008(E), as described in Chapter 4.4.3, is dictated by logical relationships between the processes and by the responsibilities for the execution of the processes.

Process constructs, as they apply to ISO/IEC15288:2008(E), as well as to ISO/IEC12207 [47], are presented in terms of a UML presentation, as shown in Figure 37.

Annex C of ISO/IEC15288:2008(E) provides the following definitions that refer to this diagram:

> A process requires a purpose and an outcome. All processes have at least one activity.
>
> Activities are constructs for grouping together related tasks. If an activity is cohesive enough, it can be converted to a (lower-level) process by defining a purpose and a set of outcomes.
>
> A task is a detailed provision for the implementation of a process.
>
> Notes are used when there is a need for explanatory information to better describe the intent or mechanics of a process.

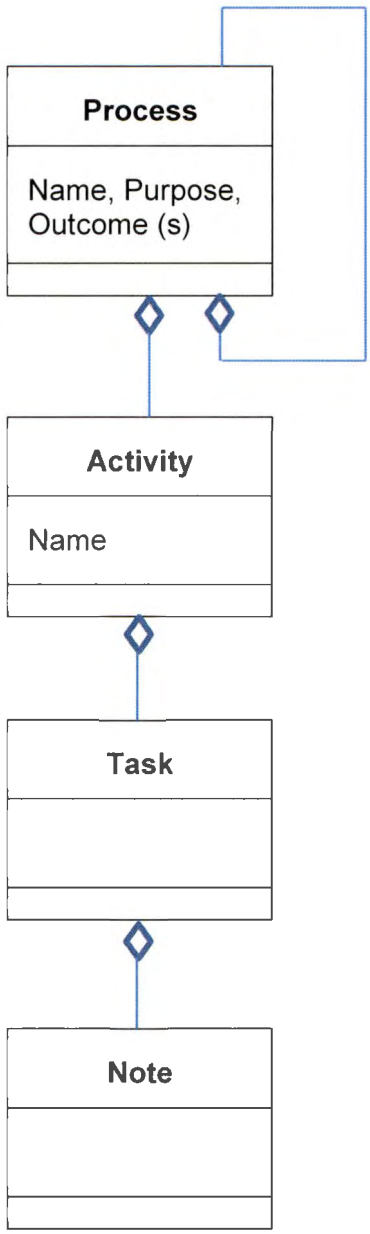

Figure 37: ISO/IEC12207/15288 Process Constructs

Estefan [43] quotes the following set of definitions regarding processes and their elements:

> "A process (P) is a logical sequence of tasks performed to achieve a particular objective. A process defines 'WHAT' is to be done, without specifying 'HOW' each task is to be performed. The structure of a process provides several levels of aggregation to allow analysis and definition to be done at various levels of detail to support different decision-making needs.

A Method (M) consists of techniques for performing a task, in other words, it defines the 'HOW' of each task. At any level, process tasks are performed using methods. However, each method is also a process on itself, with a sequence of tasks to be performed for that particular method.

A Tool (T) is an instrument that, when applied to a particular method, can enhance the efficiency of the task; provided it is applied properly and by somebody with proper skills and training. The purpose of the tool should be to facilitate the accomplishments of the 'HOWS'. In a broader sense, a tool enhances the 'WHAT' and the 'HOWS'."

Estefan [43] then describes a methodology, based on these definitions, as the application of related processes, methods, and tools to a class of problems that all have something in common.

Note that these definitions do not correspond closely with the definitions presented in Annex C of ISO/IEC15288:2008(E). However, the breakdown of processes into constituent elements follows a similar logic. It can be argued that a tool, as defined by Estefan [43], is the same as an activity, defined in ISO/IEC15288:2008(E). It can also be argued that a method, as defined by Estefan [43], is the same as a task, as defined in ISO/IEC15288:2008(E). The definitions developed below will be applied in the process design for the purposes of this study.

A widely accepted practice for describing processes, whether executed by humans or machines, in functional terms, is IDEF0. The name derives from "ICAM definition for functional modelling", where ICAM is an acronym for "Integrated Computer Aided Manufacturing". The IDEF0 formulation is detailed in the Air Force Wright Aeronautical Laboratory document ADB062457 [45]. The methodology is used to a great extent in the INCOSE Systems Engineering Handbook [48]. The IDEF0 model building block expresses a process as a function or collection of functions that produces outputs from a set of inputs, using the indicated mechanisms and controlled by the designated controls, as shown in Figure 38.

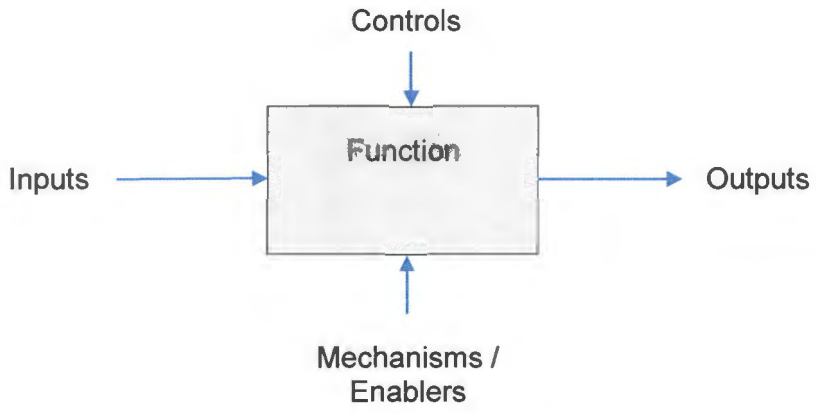

Figure 38: IDEF0 building block

The essential elements of a development process are identified in the INCOSE handbook [48] and are described in terms of "context diagrams". An interpretation of this description is shown in Figure 39.

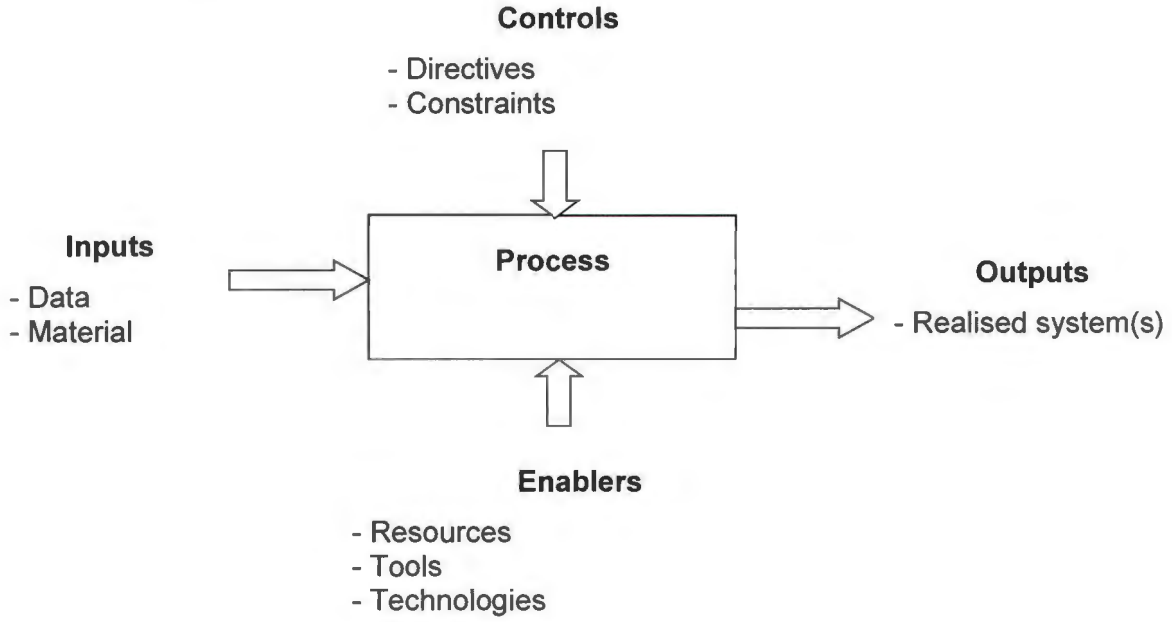

Figure 39: Development Process Concept (Adapted from Fig 1.2 in the INCOSE Handbook [48])

In this view, "a process is an integrated set of activities that transforms inputs into desired outputs" [48]. Given the notion that a (mathematical) function transforms inputs to outputs, as well as the concept presented by the IDEF0 notation, the term activity,

as described in the preceding paragraphs, can be replaced by process function. A process can thus be construed as to consist of concatenated process functions. A process function can be represented as shown in Figure 40.

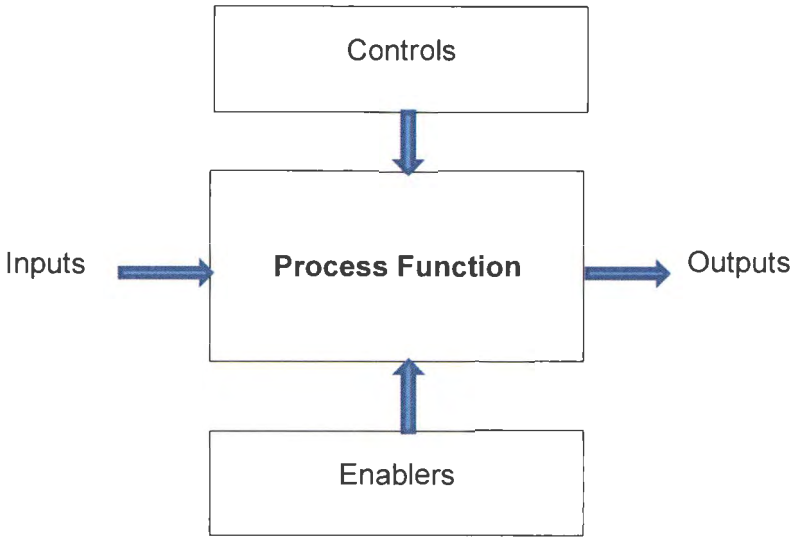

Figure 40: Process function building block

A process function consists of a set of tasks (as per the definition provided by ISO/IEC15288:2008(E) [39]).

4.4.10 Contemporary systems engineering processes - conclusions

A considerable resource for the accumulation of the collective systems engineering and software body of knowledge is found in a number of contemporary systems and software engineering standards. As part of the DSR rigour cycle these standards have been reviewed to identify development process aspects relevant to the development of complex systems containing embedded software, which could be employed in the design of a process for the development of such systems.

4.4.10.1 General

A significant difference between the classical standards and the contemporary standards is that the former typically presented detailed instructions to which the contractor had to conform, whereas the latter present more generalised descriptions of process requirements and desired outcomes, or, "what" rather than "how". This generalised approach allows the contacting enterprises to optimise its development processes to suit the specific application.

This depiction of process requirements, rather than prescribing the process to be followed, underlines the problem statement of the study, i.e. to synthesise and validate a development process specific to the stated application.

It is further observed that the contemporary standards display generic similarities in their descriptions of development process aspects, but the low level details regarding process specifics differ significantly between the standards reviewed in the study. The elements of the contemporary standards which are considered in the process design are summarised below, in terms of the research challenges formulated in Chapter 3.

4.4.10.2 Scope of engineering activities

The partitioning between project management and technical processes as described by EIA-632 [12], ISO/IEC 15288 [39] and IEEE/EIA 12207 [47], provides guidance in terms of the activities that constitute the engineering process. The information from these standards will be applied in the development of the framework for the establishment of the context of the engineering processes, in section 4.9.1.1.

The concept of a "system of systems", where the definition of a system takes the same form at any layer of the system hierarchy is clearly described in EIA-632 [12] and in ISO/IEC 15288 [39], and is specifically applied in terms of a system containing software, in IEEE/EIA 12207 [47].

The model by which a system-of-interest is decomposed into a system and enabling systems, described in EIA-632 [12], provides guidance in terms of the approach in which a system can be broken down into elements, where the generic process can be applied to the development of each element.

These three aspects, i.e. the partitioning of technical processes within a project environment, the concept of a "system of systems", and the decomposition of a system-of-interest into a system and enabling systems, will be used in the development of the process scope definition, presented in section 4.9.1, and in the definition of the context of the development process for airborne electronic equipment, described in Chapter 5.1.

4.4.10.3 Definition of activities and tasks

The engineering activities described by the contemporary systems and software standards reviewed in this section are summarised in Table 6. The activity titles in the

first column of Table 6 are defined in order to facilitate a comparison between generically similar activities as they are presented in the referenced standards.

Table 6: Engineering activities described in contemporary standards

Reference / Activity	EIA-632	ISO/IEC-15288	IEEE-1220	ISO/IEC-12207
Requirements processes	Requirements management Requirements validation	Stakeholder Requirements definition Requirements analysis	System definition stage Develop system specifications (Requirements analysis Requirements validation)	System / software requirements analysis
High-level design	Solution definition	Architectural design	Preliminary design stage Subsystem definition	Architectural design
Low-level design	Implementation	Implementation	Detailed design stage Detailed subsystem definition	Detailed design
Realisation			FAIT stage Realisation of system elements, integration and test	Software coding and testing
Integration		Integration		Integration
Verification	Verification	Verification		Qualification testing
Validation	End product and enabling products validation	Validation	(Not explicitly described)	Described as a supporting process

An observation that can be made from an assessment of Table 6 is that, although there are superficial similarities between the tasks and activities described by the standards reviewed in this section, the detailed definitions of the activities differ significantly. This observation corroborates the second research challenge in that there is no universally accepted description of activities and tasks required for the engineering of systems.

- Requirements processes

In all the contemporary standards reviewed in this section, requirements processes are stipulated which have the objective of specifying the necessary attributes of the system to be developed.

In broad terms all the standards indicate that the requirements must be analysed and guidance for the analyses processes is presented to various levels of depth in the different standards.

All the standards also allude either explicitly or implicitly to the validation of the requirements, i.e. it must be determined that the requirements are correct, and where applicable, are traceable to users' requirements.

- High-level design

All four standards also describe a high level design process, which produces a conceptual design or define a system architecture. This process defines system elements and the interfaces between the elements.

An association between the requirements processes and the high level design process is that the system requirements are allocated to the system elements.

- Low-level design

EIA-632 and ISO/IEC 15288 combine detailed design activities with other activities which are required to produce a system ready for verification, but IEEE-1220 and ISO/IEC 12207 specifically identify a detailed design activity. The objective of this activity is to design the details of the system elements identified by the high level design activity, e.g. printed circuit boards and software algorithms.

- Realisation

IEEE-1220 and ISO/IEC 12207 also specify specific activities with the purpose of fabricating the system elements according to the detailed design or to produce software source code which implement the algorithms from the detailed design process. The other standards combine this activity with "implementation".

- Integration

The integration activity is detailed in ISO/IEC 15288 and in ISO/IEC 12207. During this activity the system elements or software modules are integrated to produce the entity which is to be subjected to verification. EIA-632 considers integration as part of the implementation process, whereas IEEE-1220 deals with integration as part of the FAIT stage.

- Verification

EIA-632 and ISO/IEC-15288 stipulate a verification process where it is verified that the representative item complies with the requirements. ISO/IEC 12207 describes qualification testing, (sub-clauses 5.3.9 and 5.3.11), which follows software and system integration, as tests which are executed to demonstrate that a software product meets its specifications and is ready for use in its target environment. ISO/IEC15288:2008(E) [39] describes a verification process (sub-clause 6.4.6) which is also performed after completion of the integration process, which demonstrates compliance to specified design requirements. As the verification tasks implied by ISO/IEC15288:2008(E) are not limited to tests, i.e. it can include analyses and reviews, the verification process described by this standard encompasses a wider scope than the qualification tests described by ISO/IEC 12207. However, it is evident that the intention of this technical process is the same as the qualification tests described by ISO/IEC 12207, i.e. to provide evidence of compliance to the system and software requirements. IEEE-1220 does not distinguish clearly between tests which are performed in support of the integration process, or tests which are performed to demonstrate compliance to requirements.

It is important to understand that the item which is subjected to verification activities is representative of the product which is to be released to service. The verification results are normally used to provide evidence that the outcomes of the development process concur with contractual or certification stipulations. Therefore there should be no differences between the test item and items delivered to customers in accordance with an agreement.

- Validation

EIA-632 and ISO/IEC-15288 both describe validation processes. The purpose of these validation processes is to determine if the system, as built, is suitable for its intended use. It is evident that this activity can only be accomplished when the system is deployed in its operational environment.

Note that, if the development process was executed diligently up to this point, the system will comply with the requirements against which it was developed. Therefore, if shortcomings are identified during the validation process, it points to deficiencies in the original requirements. If no shortcomings are identified, the validation process thus served to indicate that the original requirements were correct.

4.4.10.4 Development process framework

An inspection of the life cycles described by the standards reviewed in this section and as depicted in Figure 31, Figure 33, Figure 34 and Figure 36, shows general similarities with regards to the interpretation of the development life cycle by the different standards.

The general structure of the life cycle models described by the contemporary standards reviewed in this section will be considered in the development of a life cycle model for the development of airborne electronic equipment, along with other life cycle models which are investigated in the remainder of this chapter.

The notion of a "Process Function", as shown in Figure 40, which is derived in this section, provides a convenient method by which the elements of a development process, which are generally described as processes by ANSI/EIA-632 and IEEE-1220, and as activities, by ISO/IEC-15288 and ISO/IEC-12207, can be depicted.

The Process Function concept will form a basis for further development of a life cycle model for the development of airborne electronic equipment.

4.4.10.5 Development process controls

Aspects addressed by the standards reviewed in this section which can be construed to be process control mechanisms are summarised below.

The use of configuration management and quality assurance/management processes to control the integrity of life cycle data associated with the definition of a system is described by ISO/IEC-12588 and by ISO/IEC 12207. IEEE-1220 indicates that control of process activities should be obtained through the assessments of process activities against policies and procedures maintained by the enterprise and a similar approach is described by ANSI/EIA-632, which asserts that the development process should be controlled against plans which were developed for the purpose. It will be shown in section 4.5 that this also equates to the quality assurance process.

These notions are further investigated in the next sections.

4.4.11 Contemporary systems engineering processes - summary

4.4.11.1 Summary of salient aspects

- The references describe generic lifecycle concepts;
- A separation of technical and project objectives is promoted;

- A System-of-systems approach is advocated;
- The focus is on “how” rather than “what”;
- The concepts of “end products” and “enabling products” are introduced.

4.4.11.2 Summary of concepts applied / discarded

Apply:

- The separation of technical and project objectives;
- System-of-systems approach;
- A focus on “what” rather than “how”;
- End product and enabling product concepts;
- Aspects from descriptions of task details;
- Use the references in the validation of the process design.

4.5 Quality management considerations

An organisation such as Denel Aviation is audited against applicable quality management standards on a regular basis, for accreditation purposes. For this reason, the standards against which this accreditation is conducted are reviewed in this chapter.

Quality management principles are employed by organisations to ensure the consistency of their products and services. Quality management topics have been the subject of numerous studies, and quality management related articles presenting different opinions and approaches to the problem of quality management are widely published.

In order to provide a consistent approach throughout the industry, a set of quality management standards (the ISO 0000 family) was developed by the International Standards Organisation (ISO) in 1987, defining a methodology by which an organisation can consistently provide products that meet customer and applicable statutory and regulatory requirements. This family of standards is periodically reviewed by the ISO, to remain abreast of changes in the industrial environment where these standards are relevant.

4.5.1 Quality management concepts

ISO 9000-2005 [28] defines quality management as the "co-ordinated activities to direct and control an organisation with regard to quality." The standard defines quality as "the degree to which a set of inherent characteristics fulfils requirements." In the context of this definition the requirements are needs or expectations which are stated explicitly or which are generally implied.

Organisations manage the assurance of the quality of their products and services through the employment of a Quality Management System (QMS). ISO 9000-2005 [28] describes the fundamentals of such quality management systems, as well as the vocabulary used in the ISO 0000 family of standards. The standard identifies eight principles by which improved performance of the organisation can be achieved. Two of these principles directly applicable to the problem addressed in this study are the adoption of:

- A process approach: A desired result is achieved more efficiently when activities and related resources are managed as a process (clause 0.2 d);

- A systems approach to management: Identifying, understanding and managing interrelated processes as a system contributes to the organisation's effectiveness and efficiency in achieving its objectives (clause 0.2 e).

The process approach is defined as "the systematic identification and management of the processes employed within an organisation and particularly the interaction between such processes."

Importantly, this systematic identification of processes and their mutual interactions, and considering the issue from a systems perspective in terms of the engineering of products as explicated above, form the basis of this study.

ISO 9000-2005 [28] emphasises the documenting of processes. Clause 2.7 of the standard states that documentation enables communication of intent and consistency of action. Types of process documentation which are pertinent to this study are:

- Quality manuals, which describes the Quality Management System (QMS);
- Quality plans, which describe how the QMS is applied to a specific product or project;
- Procedures, works instructions and drawings, which provide information about how to perform activities and processes consistently;
- Records, which provide objective evidence of activities performed or results achieved.

The process description which is the "artefact" which was developed in this study was developed to comply with this documentation requirement. The process description, of which a summary which is presented in Chapter 5, is described in detail in a quality manual which forms part of the Denel Aviation QMS. This process description also references procedures and works instructions which are managed through the QMS. Furthermore, the works instructions include guidance for the preparation of project specific quality plans.

The standard further requires that the QMS, describing relevant processes in terms of manuals, plans, policies, procedures and works instructions, is subjected to regular evaluations, audits and reviews to evaluate the suitability, adequacy, effectiveness and efficiency of the processes and to adapt to changes in the engineering environment. This continual improvement process forms one of the cornerstones of quality management.

In a simplified view, the quality management process can be summarised to consist of the following elements:

- Identify the processes required by the organisation;
- Formulate the processes in a documented quality manual;
- Baseline and control changes to the process documentation;
- Perform regular internal audits to confirm adherence to the processes;
- Perform regular reviews of the quality management system to ensure its applicability, and apply improvements when the need for them is identified.
- Records of activities to provide evidence of compliance with requirements as well as evidence of conformity to the processes must be kept and be available for scrutiny by external auditors, for accreditation purposes.

Importantly, it must also be pointed out that by implementing the elements of the quality management system which involves the engineering of systems, the study objective, i.e. the establishment of a documented process, as formulated in the problem statement, is achieved.

4.5.2 SAE AS9100

Enterprises worldwide are certified by accreditation bodies against the current edition of ISO 9001 [28], in terms of QMS suitability for the purpose as well as the adherence to the QMS. For organisations specifically producing products for use on board aircraft, the accreditation audits are conducted against SAE AS9100 [27]. This standard is an equivalent of ISO 9001, but contains additional material appropriate to the aerospace industry. SAE AS9100 [27] (sub-clause 7) specifies particular aspects of the product realisation process which must be complied with. These aspects are of particular interest to this study and points specifically associated with technical processes or its interfaces are summarised below.

- Planning

Sub-clause 7.1 of the standard considers the planning of the process needed for product realisation. The standard includes the planning of the quality objectives and requirements as part of the planning process, i.e. the establishment of these are not included in the requirements processes which are described in sub-clause 7.2 and 7.3. The standard lists the following aspects as quality objectives and requirements for the product:

- Product and personal safety;
- Reliability, availability and safety;
- Producibility and inspectability;
- Suitability of parts and materials used in the product;

- Selection and development of embedded software;
- Recycling or disposal of the product at the end of its life.

Furthermore, the processes and lifecycle data (the standard refers to documentation) as well as the resources to be provided which are specific to the product need to be identified and planned. It is also important to plan the required verification, validation, monitoring, measurements, inspection and test activities specific to the product; and the criteria for product acceptance must be defined. The planning process must include the identification of the records needed to provide evidence that the realisation processes and resulting product meet requirements, and must also include the identification of configuration management processes appropriate to the product. The resources to support the maintenance and use of the product must also be planned.

The project management activities (sub-clause 7.1.1) related to product realisation "shall be planned and managed in a controlled manner to meet requirements at an acceptable risk, within schedule and time constraints."

Risk management (sub-clause 7.1.2) processes shall be planned and include "assignment of responsibilities for risk management; definition of risk criteria (likelihood, consequences, risk acceptance); identification, assessment and communication of risks throughout product realisation, identification, implementation and management of actions to mitigate risks that exceed the defined risk acceptance criteria, and acceptance of risks remaining after implementation of mitigation actions."

The configuration management (sub-clause 7.1.3) activities are also to be planned. In this instance the standard references ISO 10007 [30], which is addressed in Chapter 4.6.

- Requirements processes

Requirements related processes are discussed again in sub-clause 7.2 under the heading Customer-Related Processes. It is indicated that the organisation shall determine the requirements specified by the customer, as well as those requirements not stated by the customer but which are necessary for the intended use of the product, (these requirements are referred to as "derived requirements" in the standards reviewed in the preceding section) where these are known. Statutory and regulatory requirements, and other requirements which are applicable to the product, shall be determined as well.

The organisation shall conduct a review of these requirements before the organisation commits to supply the product to a customer. The purpose of the review is to ensure

that the product requirements are defined, risks (e.g. new technology, short delivery time frame) are identified and that the organisation will have the ability to meet the defined requirements. Records of the results of the review and actions arising from the review shall be maintained.

- Design and development

Design and development processes are itemised in sub-clause 7.3. The design and development planning directives are stated in sub-clause 7.3.1. These require that the organisation shall determine product design and development stages, as well as the review, verification and validation that are applicable to each design and development stage. The responsibilities and authorities for design and development shall also be determined and communicated. The plans, in whatever format they are presented, shall be updated to reflect changes, as the design and development process progresses.

Inputs to the design and development process (sub-clause 7.3.2) include the functional and performance requirements; applicable statutory and regulatory requirements; information derived from previous similar designs if applicable; and other requirements essential for design and development. The requirements shall be reviewed to determine whether they are complete, unambiguous and not in conflict with each other. Records of the requirements and reviews shall be maintained.

The outputs of the design and development process (sub-clause 7.3.3) shall meet with the inputs for design and development and shall provide appropriate information for purchasing or production. It shall also contain or reference product acceptance criteria. Importantly, it must specify the characteristics of the system that are essential for its safe and proper use and specify, critical items and specific actions that must be taken for these items.

At suitable stages during the design and development process, systematic reviews shall be conducted (sub-clause 7.3.4) to evaluate the ability of the design and development outputs to meet requirements, to identify problems and to propose necessary actions and to authorise progression to the next stage.

Design and development verification (sub-clause 7.3.5) activities shall be performed to ensure that the outputs of the design and development process have met the design and development input requirements.

Design and development validation (sub-clause 7.3.6) shall be performed to ensure that the product is capable of meeting the requirements for the specific application or

intended use". The standard indicates that, wherever practical, this should be done prior to delivery or implementation of the product.

Tests performed in support of verification and validation (sub-clause 7.3.6.1) shall be planned, controlled, reviewed and documented. The test plans or test specifications shall identify the product being tested as well as the resources required to perform the tests. The test objectives and conditions, the parameters to be recorded, and the acceptance criteria shall be identified. Test procedures shall describe the details of how the tests are to be conducted and how the results are to be recorded. It shall be ensured that the specimen of the product which is subjected to verification and validation tests is at the correct configuration status to be representative of the product. While conducting the tests, it shall be ensured that the requirements of the test plans and test procedures are observed and that the acceptance criteria are met.

It must be demonstrated that the product definition meets the specified requirements for all identified operational conditions. This compliance shall be captured in reports, calculations, test results and other documentation (sub-clause 7.3.6.2).

Design and development changes shall be controlled (sub-clause 7.3.7), in accordance with the configuration management process.

- Measurement, analysis and improvement

Clause 8 of the standard stipulates that the organisation shall plan and implement monitoring, measurement, analysis and improvement processes. These measurement, analysis and improvement processes are primarily aimed at evaluating customer satisfaction, which include the assurance of product conformity as well as assurance of the level of conformity of the quality management system and the project specific plans. Deficiencies identified by the measurement and analysis activities are to be addressed in terms of improvements to the product as well as to the quality management system. These improvements are also subjected to measurement and analysis and the cycle is repeated, leading to a situation of continuous improvement.

Internal audits are conducted to determine the effectiveness of the QMS and to confirm that the QMS is effectively implemented and maintained. In addition, it must be ascertained that projects are executed in conformance with the plans established according to sub-clause 7.1, i.e. the product realisation planning, including customer contractual requirements. Importantly, the audits must be performed with objectivity and impartiality. Therefore, organisational arrangements and organogram structures normally ensure the independence of the personnel who are responsible for the execution of these audits.

In practise this means that internal audits are conducted on a regular basis to provide objective evidence of adherence to plans, processes and procedures. Furthermore, all tests conducted to verify compliance of the synthesised system or elements thereof, are witnessed by representatives of the quality departments of the contractor as well as from the acquirer. In addition, all lifecycle data items produced by the development process are subjected to formal reviews, where the review process is managed by personnel delegated with the responsibility for ensuring the quality of the data item. In all instances of non-compliance, corrective action is taken to eliminate the causes of non-conformance, through appropriate processes. This results in a control mechanism which ensures that products comply with requirements and standards, and that processes meet with their objectives. The development processes described by AS9100C are schematically represented in Figure 41.

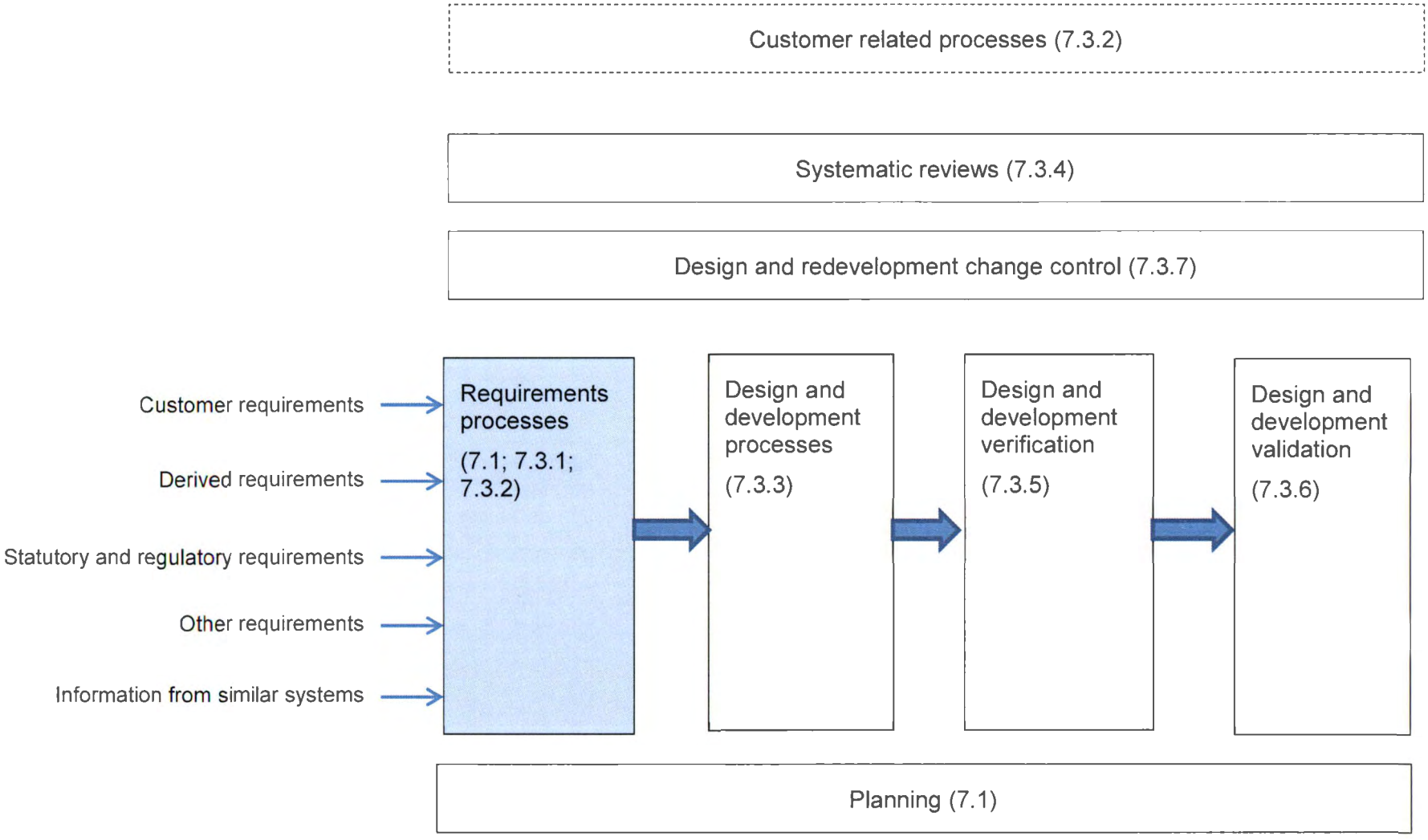

Figure 41: AS9100C development process

4.5.3 Relationship between engineering plans and the documented process

In Chapter 4.3 and 4.7 of this dissertation it is shown that a number of standards and other relevant guidance publications require the establishment of development plans for each project. One of the purposes of these plans is to communicate the processes that will be followed on the project. Note that the comprehensive definition of these processes which are described in the quality manual in the QMS serve as a generic development plan. For specific projects, the development process aspects in these plans should only refer to relevant sections in the QMS and it is hence not required to provide elaborate detail. This also ensures that different projects within the organisation will be executed in a consistent manner as they all refer to the same generic process description.

This observation is described here to provide clarity on the difference between plans which are to be generated as part of the development process, and the process description itself.

4.5.4 Synopsis of Quality Assurance References

4.5.4.1 Scope of engineering activities

The development process scope depiction in AS9100C is not significantly different from the representations described by the contemporary systems engineering standards. Technical processes are clearly identified and segregated from "customer related processes", but the processes described to control the development process are not clearly segregated between ensuring technical correctness and controls to ensure commercial objectives.

4.5.4.2 Definition of activities and tasks

Although the primary objective of the standard is to provide guidance in terms of quality, the tasks described in clause 7.3 of the standard correspond with many aspects of the systems engineering processes described by the contemporary systems engineering standards. This leads to an important observation and conclusion, described below.

Although the disciplines of systems engineering and quality management developed separately, a substantial number of convergences exists between the concepts of systems engineering as described in the preceding sections and those of quality

assurance described in this section, as both disciplines have the objective to ensure that an appropriate solution to a particular problem or need is provided by an organisation. Furthermore, systems engineering and quality management references alike promote the validation of task outcomes and the verification of realised products against requirements and standards, with the objective of meeting customer expectations.

The main difference lies in the fact that quality management is focussed on the entire organisation, whereas systems engineering is focussed on a project, or product.

This commonality of purpose is important as it contributes to a unified method of ensuring the correctness of process outcomes, as alluded to in Chapter 4.5.2, and to be discussed in Chapter 5.5.2.

4.5.4.3 Development process framework

The lifecycle model implied by AS9100C is shown in Figure 41. It is evident from this figure that the lifecycle model that can be inferred from AS9100C is fundamentally the same as the generic lifecycle model represented by the contemporary systems engineering standards, i.e. the model consists of a requirements process, which provide the inputs for the design and development processes, and against which the outputs of the design and development processes are verified. The verified product is then subjected to validation in its operational environment.

It therefore follows that the adoption of a lifecycle model which complies in basic terms with the contemporary lifecycle models will represent a development process which also conforms to the requirements of AS9100C.

4.5.4.4 Development process controls

The measurement, analysis and improvement processes described in Clause 8 of the standard relate to the improvement of the quality of both the product and the processes which produce the product. It will be shown in Chapter 4.7 that, for airworthy systems, the objective of the controls on the development process is, in addition to quality improvement, also the establishment of confidence that the product meets with its requirements and that it is safe for use with regards to its intended purpose.

It was also pointed out in Chapter 4.4.9 that the quality management processes serve as a significant element of the control processes regarding development process outputs, therefore the control activities required by the quality management process

can also be used to meet relevant objectives of systems engineering and airworthiness processes.

4.5.5 Quality management standards: conclusions

Quality management principles require the establishment of a documented engineering process definition against which the organisation can be audited for accreditation purposes.

The methods associated with the measurement, analysis and improvement process as they are defined in the quality standards can be construed to provide a means of validating development process activity outputs.

The commonality of purpose between quality management objectives and systems engineering objectives was identified.

Organisations which produce aircraft and airborne products are subjected to accreditation audits which are conducted against SAE AS9100 [27]. This standard specifies particular aspects of the product realisation process. Compliance with these aspects must be demonstrated during the accreditation audits. A review of this standard and other related standards was conducted to establish the required relationships between the development process and quality management principles.

The following aspects from the quality management standards relate to the development of airborne electronic equipment:

- Quality management principles require a systematic identification of processes and their mutual interactions and the documenting of these processes;
- The product realisation processes shall be planned and the plans shall be documented;
- The use of a well-established requirements process is mandated;
- Systematic reviews of the product realisation process shall be conducted;
- Verification activities shall be conducted to ensure that the outputs of the design and development process have met the design and development input requirements;
- Validation activities shall be performed to ensure that the product is capable of meeting the requirements for the specific application or intended use;
- Tests conducted in support of validation and verification activities shall be strictly controlled to ensure the validity of the results, and records of the results shall be kept;

- Regular audits shall be conducted to verify adherence to the documented process and project plans;
- The quality assurance process includes independent oversight over verification and validation activities to ensure the correctness of the outcomes of these activities.

Significant inferences from these aspects, which have relevance in terms of the process design are:

- The documented engineering process definition which is the outcome of this study can be used in audits for accreditation purposes;
- The methods associated with the measurement, analysis and improvement process as they are defined in the quality standards equates to a means of validating development process activity outputs with independence;
- A commonality of purpose between quality management objectives and systems engineering objectives was identified. This leads to a significant simplification in the process design as it eliminates duplication of tasks.

4.5.6 Quality management standards: summary

4.5.6.1 Summary of salient aspects

- Quality assurance principles require a process approach to ensure consistency of products and services;
- AS 9100 describes specific aspects of engineering tasks (product realisation);
- Common purpose exists between QA and SE principles.

4.5.6.2 Summary of concepts applied / discarded

Apply:

- Process approach;
- Aspects from descriptions of product realisation task details;
- Quality management as a process control mechanism.

4.6 Configuration management considerations

Configuration management was identified as an important control mechanism to be applied to the development process, in foregoing discussions in this dissertation. Configuration management is a cornerstone of engineering processes where the integrity of the product and its associated lifecycle data must be managed. As part of the DSR rigour cycle, configuration management principles are described in the following section, to provide further background to the design of the engineering process discussed in this study.

Major standards dealing with configuration management (CM) include EIA-649 [41]; and Military Handbook – Configuration Management Guidance (MIL-HDBK-61) [31]. Configuration management is also explicitly addressed in other relevant standards, e.g. RTCA/DO-178C [25]. SAE AS9100C [27], which was discussed in Chapter 0, contains specific clauses on configuration management (clause 7.1.3) and control of design and development changes (clause 7.3.7). This standard in turn invokes ISO 10007:2003 Quality management systems - Guidelines for configuration management [30].

The objective of configuration management is defined in EIA-649 [41] as to provide:

> "– The orderly establishment, documentation, and maintenance of a product's functional, performance and physical attributes
>
> – Management of changes to attributes
>
> – Access to accurate information essential to the product's development, fabrication, production, use, maintenance, procurement, and eventual disposal."

In the introduction to ISO 10007:2003 [30] it is stated that "Configuration management documents the product's configuration. It provides identification and traceability, the status of achievement of its physical and functional requirements, and access to accurate information in all phases of the lifecycle." Configuration management consist of five basic elements (ISO 10007 [30], RTCA/DO-178C [25], RTCA/DO-254 [26]), i.e.:

- Configuration planning and management
- Configuration identification
- Configuration change control
- Configuration status accounting
- Configuration verification and audits

These elements are briefly described in the following sections, and specific applicability to the control of the development process is indicated where applicable.

4.6.1 Configuration planning and management

Although the existence of an organisational CM process is a given in any organisation which is e.g. ISO 9001 compliant, configuration management should be planned for each project.

Annex A of ISO 10007:2003 [30] provides guidance on the preparation of a CM plan. This guidance details all the aspects that need to be addressed in establishing the configuration management process for a project. It must be noted that some of these aspects are project specific, but that a number of these are generic to all projects executed by the organisation, and these should be considered in the process design.

Project specific CM aspects include a description of the specific product and configuration items; project schedules; lists of project specific documents; a family tree of configuration items, specifications and other documents; configuration baselines, project specific organisational roles and responsibilities, lists of audits and their occurrence with project schedules.

Non-specific CM aspects should be embedded in the engineering processes used by the organisation. In an organisation such as Denel Aviation the aircraft level CM environment is well established and maintained, in order to support aircraft design, manufacture and maintenance processes. CM procedures and policies are defined and controlled by virtue of the corporate quality management system (QMS). As indicated above, the CM processes need to be compliant with ISO 10007:2003 [30]. In most organisations, the CM process employs and maintains automated product lifecycle management (PLM) systems to support the CM objectives of the organisation.

An important aspect of the configuration management function is the identification of a Dispositioning Authority, who is assigned responsibility and authority to make decisions on the configuration (ISO 10007 definition). At Denel Aviation Chief Designers are formally appointed per aircraft type, with the primary responsibility to review and authorise changes. The chief designers manage configuration changes through the change control process described below. Configuration management practitioners manage the CM processes, i.e. aspects such as configuration identification, baseline control and configuration status accounting.

4.6.2 Configuration identification

Principle 8 of EIA-649 [41] states that configuration identification is the basis from which the configuration of products is defined and verified; product and documents are labelled; changes are managed; and accountability is maintained.

Configuration Items and their interrelationships identify the product structure. MIL-STD-973 identified a Configuration Item (CI) as "an aggregation of hardware or software that satisfies an end use function and is designated by the government for separate configuration management." The identification of configuration items should be according to pre-defined criteria. ISO 10007:2003 [30] identifies the following selection CI criteria: statutory and regulatory requirements; criticality of items in terms of risk and safety; new or modified technology, design or development; interfaces with other configuration items; procurement conditions; and support and service. In terms of the systems which form the topic of this study, configuration items are typically hardware LRUs and SRUs, and software CSCIs.

Product configuration information describe attributes of the configuration items. This information comprises both product definition and product operational information. Product definition information typically includes requirements, specifications, design drawings, test documentation, software documentation and listings, etc. Product operational information includes maintenance and operating manuals.

Product configuration information elements should be designated by means of a numbering system or convention which uniquely identify each element (e.g. specification, design drawing) and its revision status. The policies and procedures for assigning these identifiers are typically managed in the organisation's quality management system.

A configuration baseline consists of the approved product configuration information that represents the definition of the product. Configuration baselines, plus approved changes to those baselines, represent the current approved configuration (ISO 10007). During development, baselines are used to define references for further activities in a product's lifecycle. CM Principle 16 as stated in EIA-649 [41] defines this concept as: "A baseline identifies an agreed-to description of the attributes of a product at a point in time and provides a known configuration to which changes are addressed." RTCA/DO-178C [25] defines a baseline as "the approved, recorded configuration of one or more configuration items, that thereafter serves as the basis for further development, and that it is changed only through change control procedures." "Baselines are defined for further software lifecycle process activities and allow

reference to, control of, and traceability between configuration items." The standard requires that baselines should be established for all configuration items used for certification credit.

The classical configuration identification process as described in e.g. MIL-STD-973 [32] defined three types of configuration baselines, i.e. the functional baseline (FBL) which is defined as "the initially approved documentation describing a system's or item's functional, interoperability, and interface characteristics and the verification required to demonstrate the achievement of those specified characteristics"; the allocated baseline (ABL) which describes "an item's functional, interoperability, and interface characteristics that are allocated from those of a system or higher level configuration item, interface requirements with interfacing configuration items, additional design constraints, and the verification required to demonstrate the achievement of those specified characteristics"; and the product baseline (PBL) which is "the initially approved documentation describing all of the necessary functional and physical characteristics of the configuration item and the selected functional and physical characteristics designated for production acceptance testing and tests necessary for support of the configuration item. In addition to this documentation, the product baseline of a configuration item may consist of the actual equipment and software."

According to MIL-STD-498 [10], configuration identification is concerned with the identification of entities to be placed under configuration control as well as with the configuration identification scheme to be used, including methods for indication of version/revision/release status. It is interesting to note that MIL-STD-498 [10] does not describe the concept of baselines, but mentions "build" and "version".

RTCA/DO-178C [25] requires that problem reporting starts at the establishment of the baseline at which certification credit is to be obtained. It does not identify any baseline types.

4.6.3 Configuration change control

After the initial release of product configuration information, all changes shall be controlled and the process for controlling the change should be documented (ISO 10007). The change process comprises of the formal initiation of a change, the evaluation of change requests and change proposals, the subsequent approval or disapproval of the change and the authorisation for release of lifecycle data for the change to be implemented. The change process applies to modifications to the system's design, hardware, firmware, software, and product configuration information.

In modern IT based CM processes, change process methods are associated with attributes of the employed PLM systems. After implementation of an approved change, compliance of the modified system with the change shall be verified and traceability shall be established between the verification evidence and the change details.

MIL-STD-498 [10] states that configuration control addresses the levels of control through which an identified entity must pass, identifies the persons or groups of persons who may authorise changes at different levels (including identification of acquirer responsibilities), and defines the process by which changes are requested and authorised, the tracking of changes is tracked, and past versions are maintained.

In practise this control process is effected as follows:

The archiving and retrieval of the lifecycle data items is controlled by CM practitioners. A request for change (which also includes a new development) is reviewed at a configuration control board meeting, and if the change is approved, the dispositioning authority assigned to the project sanctions the release, by the designated CM practitioner, of the appropriate lifecycle data items to the identified employees who have to perform the updates to the lifecycle data. In the case of a new document to be prepared as part of the development, it is the responsibility of the CM practitioner to ensure that the correct template for the document is issued and that the configuration identification of the document is correct. After completion of the changes to a lifecycle data item, it is subjected to the validation process as described in Chapter 0. Only when the person delegated with the responsibility for ensuring the quality of the data item is satisfied that all the quality assurance activities related to the data item are completed, this person forwards the completed data item, as well as the associated quality records, to the CM practitioner, who then archives the updated data item.

As a side observation, it must be noted that although project and resource management is excluded from this study, work control, i.e. authorisation to perform work from a resource management perspective, can be coupled to the change control process to simplify the process control mechanism.

4.6.4 Configuration status accounting

Principle 35 of EIA-649 [41] states that "an accurate, timely information base concerning a product and its associated product information is important throughout the product lifecycle." Configuration status accounting is defined in MIL-HDBK-61A [31] as "the process of creating and organising the knowledge base necessary for the performance of configuration management. In addition to facilitating CM, the purpose

of CSA is to provide a highly reliable source of configuration information to support all project activities including program management, systems engineering, manufacturing, software development and maintenance, logistic support, and maintenance."

The configuration status accounting activity results in records and reports that relate to a product and its product configuration information. Configuration status accounting records are created during the configuration identification and change control activities. These records allow for visibility and traceability and for the efficient management of the evolving configuration. These records typically include details of the product configuration information (identification number, title, effective dates, revision status, change history and inclusion in any baseline) and of the product's configuration (part numbers, design or build status) (ISO 10007).

To protect the integrity of the product configuration information and to provide a basis for the control of change, it is recommended that the configuration items and related information be held in an environment that provides protection from corruption or unauthorised change, that provides means for disaster recovery, and that permits retrieval (ISO 10007).

4.6.5 Configuration verification and audits

Configuration audits are performed in accordance with documented procedures to determine whether a product conforms to its requirements and product configuration information (ISO 10007).

ISO 10007 identifies the following configuration audits:

- A functional configuration audit is a formal examination to verify that a configuration item has achieved the functional and performance characteristics specified in its product configuration information.
- A physical configuration audit is a formal examination to verify that a configuration item has achieved the physical characteristics specified in its product configuration information.

It is important to note that the standard states that the configuration audits are not intended to replace other forms of verification. The functional and physical configuration audits were also identified by the classical systems engineering process definition as described in Chapter 4.3.2.

The objective of the FCA is to confirm that the configuration item (CI) has achieved the functional and performance requirements allocated to it. This can only be done when qualification test results concerning the CI are available.

The objective of the PCA is to verify that the "as built" item conforms to the technical documentation that identifies the item. This implies that the configuration item must be inspected against the production drawings and other related documents. This audit proves that the production lifecycle data is correct and that it was sufficient to produce the CI. This can only be done after a CI was built using the production data pack. Note that it is an AS9100 requirement that a representative item from the first production run of a new part shall be inspected against its production data pack. This is equivalent to a PCA.

4.6.6 Archiving, recovery and control of software

RTCA/DO-178C [25] indicates that the CM process provides the ability to consistently replicate the executable object code and parameter data item files, for software "manufacturing" purposes or to re-generate it in the case of an investigation. In addition, the CM process ensures that the physical archiving, recovery and control are maintained for the configuration items. Implied in these statements is the requirement that measures shall be put in place to ensure data retention, i.e. the data storage media shall be managed in terms of obsolescence and losses due to ageing and the tools required for replication of the executable object code, e.g. operating systems, compilers and computer platforms shall be maintained throughout the operational lifecycle of the system where the object code is utilised.

Also note that MIL-STD-498 [10] describes software packaging, storage, handling and delivery under the heading of configuration management (sub-clause 5.14.5), stating that the developer is responsible for the control of the storage and distribution of the deliverable software products, and to maintain master copies of the software products for the duration of the contract.

4.6.7 Control categories

Another very important notion from RTCA/DO-178C [25] is the segregation of configuration management control categories into Control Category 1 (CC1) and Control Category 2 (CC2). Lifecycle data can be assigned to either. The standard assigns a control category to each lifecycle item identified by the standard per lifecycle data item; depending on the software level.

CC1 requires configuration identification; baselines; traceability; problem reporting; change control in terms of integrity and identification as well as change control in terms of change tracking; change review, configuration status accounting; retrieval control; protection against unauthorised changes, media selection, refreshing and duplication; release and data retention.

CC2 only requires configuration identification; traceability; change control in terms of integrity and identification; retrieval control; protection against unauthorised changes, and data retention.

Typical items requiring CC1 control are requirements and design data, source code, executable object code, and trace data. Typical items requiring CC2 control are validation and verification results.

In terms of all the lifecycle data which represents the characteristics of the software, i.e. requirements, design and source code, it must be confirmed that the data is accurate, consistent and verifiable, and that the data item is in conformance with the applicable standards. It must also be shown that the source code is complies with, and is traceable to the software design and, in turn, that the design complies with, and is traceable to the requirements. This includes confirmation that the source code contains no undocumented functionality. Validation of test cases and procedures include the confirmation that test coverage of the requirements and design is achieved, and that the test cases and procedures comply with, and are robust (i.e. it was shown that the software can continue to operate correctly despite invalid inputs (RTCA 178B definition)) the requirements and design. It must also be confirmed that the test results are correct, and discrepancies explained.

4.6.8 Synopsis of Configuration Management References

The configuration management references mainly relate to tasks and activities associated with product data control processes, therefore only the research challenge concerned with the identification of appropriate development process control mechanisms is discussed below.

A shortcoming in the process at Denel Aviation, as identified in the retrospective assessment, was that no dispositioning authority was assigned on the level of e.g. the Rooivalk AFCS or Oryx AIU, which impeded the control over the development process. *Therefore, provision must be made for the indication of the role of a product or system-of-interest specific dispositioning authority.*

Baselines, based on a protocol for configuration identification, form the cornerstone for the control of the integrity of the outputs of the development process. Baselines are also used to manage the integrity of systems in operational deployment, through build and version control.

The review of configuration management processes has shown that aspects of the configuration management process can effectively be employed as development process control mechanisms providing a partial solution to the research challenge of resolving the ineffective management of the integrity of process outputs.

Two distinct roles of the CM process can be identified, i.e. to ensure that lifecycle data for a product is accurate and consistent with the physical attributes of the system during development, and secondly, to ensure the integrity of the system and its supporting information throughout its service life.

4.6.9 Configuration management references: conclusions

The configuration management process provides a suitable mechanism by which the technical aspects of the development process can be controlled and co-ordinated.

The control process is based on baselines which are established at different stages of the project, to serve as a basis for further activities in the lifecycle and facilitates reference to, and traceability between configuration items.

The different control categories provide an additional reference according to which different elements of the development process can be differentiated, which furthers the generalisation of the process.

Configuration management processes are used to ensure that the lifecycle data associated with a physical item is consistent with the attributes of the system. These processes are well established in organisations producing systems where the integrity of the system and the documentation associated with the system is essential, during development as well as through the operational use of the system. Configuration management is useful to manage integrity of the lifecycle data produced during development, therefore configuration management practises, as described in applicable standards, were reviewed in the literature study.

The principal elements of a configuration management process are:

- Configuration planning and management
- Configuration identification
- Configuration change control

- Configuration status accounting
- Configuration verification and audits

The concept of control categories was introduced. Importantly, control category 1 can be considered to apply to lifecycle data where baseline management, i.e. changes authorised by means of a configuration control board, whereas control category 2 applies to lifecycle data that can be modified without formal authorisation.

Configuration management forms a significant element of the process design, for the following reasons:

- Baselines, based on a protocol for configuration identification, form the cornerstone for the control of the integrity of the outputs of the development process. Baselines are also used to manage the integrity of systems in operational deployment, through build and version control;
- An adequately structured CM process, operating in conjunction with an efficient lifecycle data validation process as described by the quality assurance process, provides for a sufficient development control method, and no additional controls over the technical processes are required;
- Configuration status accounting by the organisation provides a means for recording the development history of the product. This information is useful to support the decision making process when changes need to introduced during the operational life of the product. For products with airworthiness approval, the visibility on the design history is important during investigations that may be conducted after a failure or incident involving the product.

4.6.10 Configuration management references: summary

4.6.10.1 Summary of salient aspects

- CM standards describe principles and methods to control the development of lifecycle data and to manage the integrity of this data.

4.6.10.2 Summary of concepts applied / discarded

Apply:

- Configuration management as a process control mechanism;
- Baseline management principles;
- Control categories to manage the CM level of effort.

4.7 Airworthiness recommended practises

This study is concerned with the development of airborne systems. Airworthiness concepts are discussed next, in order to enhance the understanding of the background to the problem of developing airworthy equipment, as well as to determine further process design parameters.

4.7.1 Recommended practise documents

When a standard, e.g. ISO/IEC 15288, is invoked by an agreement, it implies that all work performed under the agreement must comply with the standard. Non-compliance need to be negotiated between the parties to the agreement, and waivers are defined and agreed. In order to avoid the problem with managing non-compliance, the contemporary guidance publications for airworthiness including SAE ARP4754 [33], SAE ARP4761 [34], RTCA/DO-254 [26] and RTCA/DO-178C [25] are designated as "Recommended Practise" documents. These publications provide guidance on their respective topics, but for each project (development of an aircraft, system, or software) the detail of implementation of the recommended practise is agreed to between the contractor and the certification authority.

4.7.2 Airworthiness concepts

Aircraft and airborne systems are permitted to operate on a commercial basis if they are granted airworthiness certification by the relevant airworthiness authorities. The prime objective of the process by which airworthiness is granted, is to ensure that aircraft and airborne systems are safe for use.

Airworthiness is managed worldwide by statutory organisations, such as the Federal Aviation Administration (FAA) in the United States, the European Aviation Safety Agency (EASA) in Europe and the Civilian Aviation Authority (CAA) in South Africa.

A manufacturer of aircraft or airborne equipment needs to demonstrate compliance with prescribed airworthiness standards to the applicable authority for the specific country, (e.g. Federal Aviation Regulations (FARs) published by the FAA in the USA) which then issues a Type Certificate (TC) for the aircraft type to the specific manufacturer. This TC is evidence that the aircraft is authorised to be operated in the airspace of the relevant country and for commercial gain.

Materials, parts, processes and equipment not directly associated with a specific aircraft type, e.g. VHF radios and cockpit instruments, are granted a Technical Standard Order (TSO) approval, and may be used on type certificated aircraft. The standards (FARs) typically require proof of integrity of the design, adequate performance with respect to safe operation, acceptable handling qualities of the aircraft throughout the permissible operating envelope and reliability and redundancy of safety-critical subsystems.

The most noteworthy and widely used contemporary airworthiness publications are the FAA regulations, EASA regulations, RTCA airworthiness guidelines (including RTCA/DO-160 [22], RTCA/DO-178B/C [11][25], RTCA/DO-254 [26]) and SAE recommended practice documents (including SAS9100 [27], ARP 4754 [33] and ARP 4761 [34]).

The FAA distinguishes between airworthiness certification and airworthiness approval. According to SAE ARP4754 [33], airworthiness is "the condition of an item (aircraft, aircraft system, or part) in which that item operates in a safe manner to accomplish its intended function". Airworthiness certification denotes "the processes to obtain legal recognition from authorities that the aircraft is safe to operate in its intended role" and airworthiness approval is understood to be "the processes to obtain legal approval from authorities that a system is acceptable for use on board an aircraft for which a valid airworthiness certificate is issued."

The certification authorities apply the term "certification" to aircraft, engines and propellers; and the term "approval" is used for all other systems.

An important concept underlying the airworthiness approval process is the certification basis. The following definition is extracted from RTCA/DO-254 [26]: The certification basis is "defined by the Certification Authority in consultation with the Applicant, as the particular certification requirements, together with any special conditions which may supplement the published regulations that become the basis for certification of the aircraft, engine, or propeller."

Closely associated with the certification basis is the concept of the means of compliance. RTCA/DO-178C [25] expresses the means of compliance as "the intended method(s) to be used by the applicant to satisfy the requirements stated in the certification basis for an aircraft or engine. Examples include statements, drawings, analyses, calculations, testing, simulation, inspection and environmental qualification."

The means of compliance with respect to the certification basis is agreed to between the certification authority, and the applicant.[5]

Certification of an aircraft as well as approval of the systems used on board the aircraft, is granted by the designated airworthiness authority, based on information provided by the applicant. For an aircraft, the manufacturer produces a Certificate of Design (CoD), which references compliance to the certification basis, and after scrutiny of the CoD the airworthiness authority will grant a Type certificate (TC).

The development process and lifecycle models must therefore provide for the processes that generate this information.

The most frequently referenced recommended standard for dealing with airworthiness approval of airborne electronic equipment is Aerospace Recommended Practice (ARP) ARP4754A - Guidelines for Development of Civil Aircraft and Systems [33].

ARP 4754A identifies eight elements of the lifecycle which need to be considered when planning a project for the development of civil aircraft and systems.

- Development entails the process and methods to be used for the system architecture development, integration and implementation;
- The safety program consists of the safety activities related to the development of the system;
- Requirements management involves the capturing and management of requirements;
- Validation is concerned with showing that requirements and assumptions are complete and correct;
- Implementation verification encompasses processes and criteria which are applied to show that the implementation satisfies its requirements;
- Configuration management consists of the description of key development related configuration items and the management thereof;
- The process assurance process provides assurance that the defined practises and procedures to be applied during system development are followed;

[5] It is also significant to note that the means of compliance in terms of the contract is usually also negotiated with the contracting agency with respect to demonstration of compliance with contractual requirements.

- Certification entails the process and methods which are used to achieve certification of the system.

It is important to note, that for items for which certification are sought, the plans describing these elements are approved by the certification authority, and when it is required to deviate from these plans, these deviations also need to be approved. For the development of software, RTCA/DO-178C [25] recommends that the development process and means of compliance is described in a Plan for Software Aspects of Certification (PSAC). RTCA/DO-254 [26] advises the use of a Plan for Hardware Aspects of Certification (PHAC) for the same purpose, when development of complex hardware is concerned.

SAE ARP4754A [33] also identifies three project phases, i.e. a concept, development and production/operation phase. The document is primarily concerned with the development phase of the project. It states that development is complete when:

- "Build/test information is provided to a production facility;
- All regulatory compliance data is submitted and approved;
- The design has met all internal compliance data (as required);
- Limitations, maintenance and other operational information are provided to the aircraft operators."

In terms of the system development process, no preferred method or process, nor specific organisational structure is implied. It is acknowledged that although a top-down implementation sequence provides a convenient conceptual process model for development, typical development occurs "in an iterative and concurrent fashion using both top-down and bottom-up strategies." The guidance provided by the recommended practise is based on the top-down approach to simplify the associations with the system safety processes.

Importantly, in the case of the development of airborne electronic equipment, the system safety process forms an integral part of the development process. SAE ARP4754A [33] invokes ARP4761 - Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment [34] for guidance on the system safety process.

4.7.3 System safety

4.7.3.1 SAE ARP4761

The system safety process described by SAE ARP4761 [34] has three main elements, namely a Functional Hazard Assessment (FHA), a Preliminary System Safety Assessment (PSSA) and a final System Safety Assessment (SSA).

- Functional Hazard Assessment

ARP4761 [34] defines a hazard as "a potentially unsafe condition resulting from failures, malfunctions, external events, errors, or a combination thereof". The FHA is conducted at the onset of a project with the objective to identify and classify the failure conditions associated with the specified aircraft or system functions in terms of the severity of the hazards posed by identified functional failure conditions. An identified failure condition is classified in accordance to the severity of its effects as defined in FAA advisory circulars AC 23.1309 [35] and FAA AC 25.1309-1A [36]. FAR AC 23.1309 classifies the functional failures according to total loss of function; loss of primary means of providing the function; and misleading and / or malfunction without warning, and also links these to particular phases of flight. A summary of the formulation of the classification of hazards provided by these documents is presented below to provide a view on the FHA process.

- If the failure of the function is such that an accident with fatal consequences is unavoidable, the associated failure condition is classified as catastrophic. In this instance it must be demonstrated that the implementation of the function is such that the likelihood of the associated undesirable event to occur is extremely improbable, which is interpreted to correspond to a quantitative measure of probability of less than 10^{-9} occurrences per flight hour.
- If the failure conditions are such that the failure may lead to an accident, or that it places such a workload on the crew that their capability to perform their tasks is severely compromised, or the failure has adverse effects on the passengers, the failure condition is classified as hazardous (FAR AC 23.1309) or severe major (FAR AC 25.1309). In this case it must be demonstrated that the implementation of the function is such that the likelihood of the related event to occur is extremely remote, which corresponds to a probability figure of less than 10^{-7} occurrences per flight hour.
- If a functional failure causes aircraft safety or functional capability to be jeopardised significantly and the resulting increase in workload is such that the

efficiency of the crew is adversely affected, or the failure causes some discomfort to the passengers, the failure condition is classified as major. For major functional failure conditions, evidence must be provided to support claims that the likelihood of a failure is remote, corresponding to less than 10^{-5} failures per flight hour.

- If there is no significant reduction in safety margins or functional capabilities, the failure condition is classified as minor. It is assumed that the probability of occurrence of such a class of failure is reasonably probable, corresponding to less than 10^{-3} failures per flight hour.

The probability figures stated above are numbers generally accepted by the aerospace community and are used in the referenced standards. The FHA process is shown graphically in Figure 42.

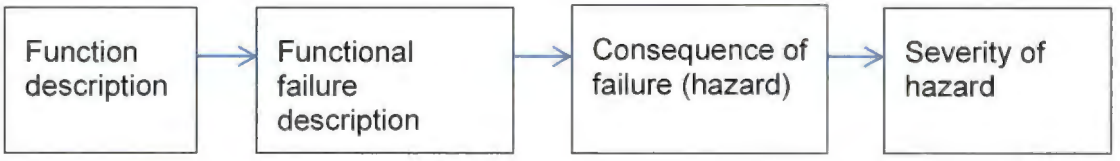

Figure 42: FHA process

The FHA results are used to determine safety objectives for the system-of-interest. The safety requirements typically translate into design constraints or limitations with respect to the use of the system. It is important to note that the FHA process should remain active throughout the development process, in order to deal with derived requirements as they are identified. It is imperative that all system functions are identified and formulated as requirements, to ensure that no intended or implied system functionality is not scrutinised by the FHA process. If additional functional requirements are identified as a consequence of the synthesis process, these requirements must be recorded and provided to the FHA process, to determine the effects of failures of these functions. This implies that the requirements process should also remain active throughout the full development cycle.

- Preliminary System Safety Assessment

At the start of the design process, the PSSA process investigates the proposed system architecture to determine how the functional failures identified in the FHA can be caused by failures of proposed system elements. The PSSA process uses techniques such as fault tree analyses, failure mode and effect analyses and Markov analyses. Hazards can be mitigated by means of architectural solutions, e.g. redundancy, monitoring or partitioning, where feasible. If practical, sufficient redundancy can be built

into the system so that the failure of a single element or subsystem still allows correct operation of the system with respect to the identified functional failure, albeit with reduced safety margins. In this case, the hazard is deemed to have been mitigated by system architecture design. If it can be shown that the contribution of a specific component to a classified failure cannot be mitigated, then it must be shown by means of testing or analyses that the probability of failure of the system corresponds to the desired likelihood of the failure as explained above. If this cannot be achieved, e.g. due to the complexity of the system, then the component or software must be developed at a development assurance level which is commensurate with the criticality of the hazard.

When the development process is completed, the SSA is conducted to gather evidence that the system-of-interest, as designed and realised, is safe with respect to its intended use. The SSA process uses the same type of analyses than what was used in the PSSA process, but with the objective to verify that the safety objectives were met.

In summary: the system safety process requires that the functional requirements (including derived requirements) for a system are defined unambiguously and comprehensively. These are assessed in terms of their associated failure conditions, to derive safety objectives for the system. The system design as well as design processes are influenced by these safety objectives and the final implementation is subjected to verification processes where compliance with the safety objectives and design and software assurance processes are demonstrated.

4.7.3.2 Other System Safety References

Although the recommended practises described above provide comprehensive guidance on the processes to ensure the safety of airborne systems, other system safety standards used by military contracting agencies are described below, for further background information. It is important to understand the underlying principles expounded by these standards as they are widely referenced in terms of the development of equipment for military use.

MIL-STD-882E: Department of Defence Standard Practise – System Safety [37] "identifies the DoD approach for identifying hazards and assessing and mitigating associated risks encountered in in the development, test, production, use, and disposal of defence systems." It is important to note that this standard applies to a much wider class of systems which include all types of military equipment.

The system safety process described by the standard consists eight elements summarised below.

1. Document the system safety process

It must be described how risk management is integrated into the systems engineering process and into the development lifecycle, as well as in the programme management structure. All system requirements must be identified and documented. The method for acceptance of hazards and associated risks by the appropriate authority must be formulated and recorded. Identified hazards must be documented and a closed loop Hazard Tracking System (HTS) must be implemented. This HTS should include at least the following: identified hazards, associated mishaps, risk assessments, identified risk mitigation measures, hazard status, verification of risk reductions, and risk acceptances.

2. Identify and document hazards

The standard defines a hazard as "a real or potential condition that could lead to an unplanned event of series of events (i.e. mishap) resulting in death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment. A systematic analysis process which includes hardware, software, interfaces (including human-machine interfaces), and intended use and operational environment is to be used for the identification of hazards. The entire lifecycle of the system shall be considered and potential impacts on personnel, the public and the environment should be assessed. Identified hazards shall be documented in the HTS.

3. Assess and document risk

The severity category and probability level of potential mishaps for each hazard across all system modes shall be assessed. The standard defines severity categories and probability levels which are to be used in the assessments. The severity categories are summarised in Table 7, and the probability levels are summarised in Table 8.

The standard also provides a risk assessment matrix where the severity of a mishap is paired with the probability of the mishap. This matrix assigns risks to be classified as high, serious, medium or low, according to severity and probability of occurrence. These risks are to be documented in the HTS.

Table 7: Severity categories (MIL-STD-882E)

Description	Severity Category	Mishap Result Criteria
Catastrophic	1	Could result in death, permanent total disability, irreversible significant environmental impact, or monetary loss equal to or exceeding $ 10 M.
Critical	2	Could result in permanent partial disability, injuries or occupational illness that may result in hospitalisation of at least three personnel, reversible significant environmental impact, or monetary loss equal to or exceeding $ 1 M but less than $ 10 M.
Marginal	3	Could result in injury or occupational illness resulting in one or more lost work day(s), reversible moderate environmental impact, or monetary loss equal to or exceeding $ 100 K but less than $ 1 M.
Negligible	4	Could result in injury or occupational illness not resulting in a lost work day minimal environmental impact, or monetary loss less than $ 100 K.

Table 8: Probability levels (MIL-STD-882E)

Description	Level	Specific Individual Item	Fleet or Inventory
Frequent	A	Likely to occur often in the life of an item.	Continuously experienced.
Probable	B	Will occur several times in the life of an item.	Will occur frequently.
Occasional	C	Likely to occur sometime in the life of an item.	Will occur several times.
Remote	D	Unlikely, but possible to occur in the life of an item.	Can reasonably be expected to occur.
Improbable	E	It can be assumed it may not occur in the life of an item.	Unlikely to occur, but possible.

4. Identify and document risk mitigation measures

Potential risk mitigations are to be identified and the expected risk reductions of the alternatives must be estimated and recorded in the HTS. The standard describes the following process to be followed in this regard:

> Hazards are to be eliminated through design selection by e.g. selecting a design or material that eliminates the risk.
>
> If the risk cannot be mitigated by design or material selection, design changes which may reduce the severity and/or probability of the mishap potential must be considered.
>
> If design alteration is not feasible, the severity or probability of the mishap should be reduced by means of engineering features or devices, e.g. a device that actively interrupts the mishap sequence.
>
> If engineering features are not feasible or deemed effective, detection and warning systems should be incorporated into the design to alert personnel to the presence of a hazardous condition or the occurrence of a hazardous event.
>
> If none of the above approaches eliminate the hazard, then signage, procedures, training and use of personal protective equipment should be considered.

5. Reduce risk

Mitigation measures identified by the approach described above are selected and implemented to achieve an acceptable risk level.

6. Verify, validate and document risk reduction

The implementation of risk mitigation measures must be verified and its effectiveness must be validated. The verification and validation results must be recorded in the HTS.

7. Accept risk, and document

The risks shall be accepted by the appropriate authority, according to a prescribed procedure, before the system is operationally deployed.

8. Manage life-cycle risks

After operational deployment of the system, the system safety process shall be used to identify hazards and maintain the HTS throughout the lifecycle of the system.

The standard recognises that the assessment of risk as far as software is concerned cannot only rely on the risk severity and probability. The standard describes a different approach for the assessment of the contribution of software to the system risks.

The system safety hazard analysis and the software system safety hazard analysis processes identify the exact software contributors to hazards and mishaps. The standard (sub-clause 4.4.1 b) states that "the successful execution of pre-defined level of rigour tasks increases the confidence that the software will perform as specified to software performance requirements, while reducing the number of contributors to hazards that may exist in the system.

The software assessment process is based on the concept of "software control categories". These are summarised in Table 9.

The standard presents a matrix for pairing the software control category with the severity category of the identified mishaps. Whereas the risk assessment matrix resulted in risk levels, the software safety criticality matrix produces software criticality indexes (SwCIs). The SwCIs are associated with the level of rigour tasks which are required to provide confidence that the system level risks are mitigated. The level of rigour tasks associated with the SwCI categories are summarised in Table 10.

The standard provides further guidance in terms of the management of risk, based on SwCI, risk level, the status of level of rigour tasks, and risk assessments and acceptance. This guidance is not relevant to the development process per se and is not discussed further in this study.

A comparison between the system safety process described by MIL-STD-882E [37] and that described by SAE ARP4761 [34] show a similar approach; however, the concepts described by MIL-STD-882E [37] will not be implemented in the development process described in Chapter 5.

Table 9: Software control categories (MIL-STD-882E)

Level	Name	Description
1	**Autonomous (AT)**	Software functionality that exercises autonomous control authority over potentially safety-significant systems without the possibility of predetermined safe detection and intervention by a control entity to preclude the occurrence of a mishap or hazard.
2	**Semi-Autonomous (SAT)**	Software functionality that exercises control authority over potentially safety-significant systems allowing time for predetermined safe detection and intervention by independent safety mechanisms to mitigate or control the mishap or hazard.
3	**Redundant Fault Tolerant (RFT)**	Software functionality that issues command over potentially safety-significant systems requiring a control entity to complete the command function.
4	**Influential**	Software generates information of a safety-related nature used to make decisions by the operator, but does not require operator action to avoid a mishap.
5	**No safety impact (NSI)**	Software that does not possess command or control authority over safety-significant systems and does not provide safety-significant information.

Table 10: Software level of rigour tasks (MIL-STD-882E)

SwCI	Level of rigour tasks
SwCI 1	Program shall perform analysis of requirements, architecture, design, and code, and conduct in-depth safety-specific testing.
SwCI 2	Program shall perform analysis of requirements, architecture, and design, and conduct in-depth safety-specific testing.
SwCI 3	Program shall perform analysis of requirements and architecture, and conduct in-depth safety-specific testing.
SwCI 4	Program shall conduct safety-specific testing.
SwCI 5	None required.

Defence Standard 00-56 – Safety Management Requirements for Defence System [51] issued by the Ministry of Defence of the United Kingdom, specifies requirements for the management of safety, to be implemented by contractors to this ministry.

This standard makes reference to the notion of a safety case, which is "a structured argument, supported by a body of evidence that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given operating environment."

The safety case therefore provides a summary of the reasoning that was followed to establish confidence that the system is safe for use. The standard does however not describe the methods to be used to establish the safety case, to the extent which is found in SAE ARP4761 [34].

Considering the fact that the SSA from the SAE ARP4761 [34] process presents the verification of the safety requirements, it can be argued that the safety case serve the same purpose as the SSA.

4.7.4 RTCA/DO-178C

Due to its prominence in the development of airborne electronic equipment, the recommended practise for development of embedded software, RTCA/DO-178C, is examined in the next chapter. It was shown in a preceding chapter that MIL-STD-498 was extensively prescriptive in terms of the types, format, and contents of the documentation to be provided during the development of a system containing embedded software. This standard could have been described as a process design. As mentioned, this standard was cancelled in 1998, and the authorities recommended the use of RTCA/DO-178B (issued in 1992) as a certification basis. A significant difference between these standards is that DO-178B specifies development process requirements, rather than presenting a process design. The standard was updated and issued at revision C in December 2011.

RTCA/DO-178C [25] provides two definitions for the term "lifecycle", i.e. "(1): An ordered collection of processes determined by an organisation to be sufficient and adequate to produce a product. (2): The period of time that begins with the decision to produce or modify a product and ends when the product is retired from service."

This standard identifies three types of lifecycle processes:

- The software planning process, where the activities of the software development and integral processes are defined;

- The software development process, which produces the software product. The standard identifies the software requirements process, the software design process, the coding process and the integration process as sub processes.
- The integral processes, "... ensure correctness, control, and confidence of the software lifecycle processes and their outputs". These include software verification, configuration management, and quality assurance and certification liaison. The standard also emphasises that these integral processes are performed concurrently with the development processes.

Essential aspects of the standard are summarised below.

- Planning

The outputs of the planning process, notably the "Plan for Software Aspects of Certification (PSAC)" are approved by the airworthiness authority, affirming that the intended lifecycle will be sufficient to meet with the software airworthiness objectives. The standard also requires the establishment of internal standards, e.g. requirements -, design - and coding standards, to provide a basis against which outputs of tasks can be validated. Development activities are audited to confirm compliance with the plans and internal standards, and the resulting audit reports form part of the process assurance evidence. In other words, plans are used as references against which the quality of work is assessed, and not purely to communicate development details with stakeholders.

- Development, validation and verification

Software development activities are identified to comprise of software requirements-, design-, coding- and integration processes. The software design process is broken down into software architecture and software detailed design activities.

The standard promotes requirements based verification testing. Three types of *verification* tests are identified, i.e. hardware/software integration testing, software integration testing (not to be confused with testing during the integration phase) and low-level testing. Hardware/software integration testing verifies correct operation of the software in the target computer environment. These tests are typically performed in test rigs, simulators or during flight tests. Software integration testing is aimed at the verification of the implementation of the software requirements and components in the software architecture. These tests are generally performed on the host computer. Low-level testing verifies correct operation of the software low-level requirements, e.g. detailed algorithm design aspects. The standard points out that low-level testing should

only be performed if test coverage of the low-level requirement cannot be achieved through software integration testing or hardware/software integration testing. The following is extracted directly from the standard (clause 6.4): "To satisfy the software testing objectives:

a) Test cases should be based primarily on the software requirements.

b) Test cases should be developed to verify correct functionality and to establish conditions that reveal potential errors.

c) Software requirements coverage analysis should determine what software requirements were not tested.

d) Structural coverage analysis should determine what software structures were not tested."

The standard requires that verification and validation is performed with independence, i.e. the outputs of a task cannot be verified or validated by the person(s) who produced the outputs, dependent on the software level, as defined in the preceding section.

- Development lifecycle model

Annex A of RTCA/DO-178C presents guidelines for the software lifecycle objectives and outputs as identified in tabular form, which provides checklists against which development activities can be assessed. The tables list development tasks identified in the document on a point by point basis, referring to the contents of the standard and indicate the applicability of the task and independence requirements with respect to verification and validation activities. They also list outputs of the development tasks associated with the selected software level.

In order to improve understanding of the essence of the standard, a diagrammatic depiction of the contents of these tables was derived as part of this study to provide an illustrative view of the software development and associated verification and validation processes and interactions between them. This diagrammatic representation of the standard in terms of tasks, outcomes and validation and verification activities is shown in Figure 43. The diagram shows the main developmental activities and expected outcomes, as well as the verification and validation activities, including the requirements for independence of validation and verification with respect to software levels.

The diagram below does not appear in the standard, but is necessary to show the main development process activities and their interrelationships and dependencies. From the diagram, it is seen that there are four main activities, namely requirements

development, software design and implementation, and integration and testing. These activities are further abstracted further to simplify the development process model.

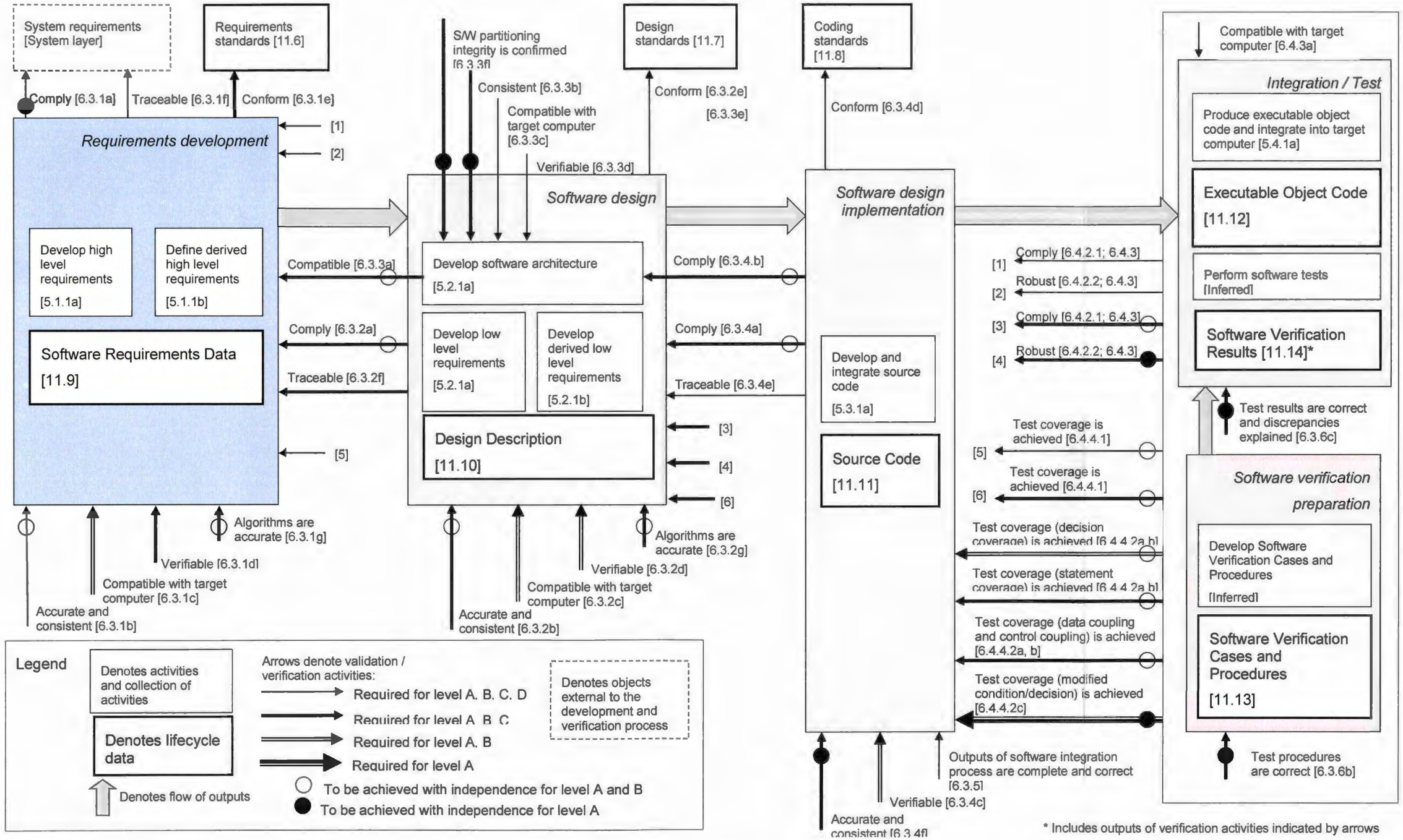

Figure 43: RTCA/DO-178B Software Development, verification and validation processes

4.7.5 RTCA/DO-254

Following on the introduction of RTCA/DO-178B, RTCA/DO-254 - Design Assurance Guidance for Airborne Electronic Hardware [26] was issued to provide guidance for electronic hardware, particularly with reference to "complex" electronic hardware, such as field programmable gate arrays (FPGAs) and programmable logic devices (PLDs).

As can be inferred from its title, the objective of RTCA/DO-254: Design Assurance Guidance for Airborne Electronic Hardware [26], is to provide guidance on ensuring the integrity of the design of airborne electronic equipment throughout its entire lifecycle, in terms of airworthiness. This is achieved through the use of the principle of design assurance.

The RP identifies the following hardware design lifecycle processes:

- The planning process, which defines and co-ordinates the activities for the project;
- Design processes, which generate the design data and the resultant hardware item;
- Supporting processes, which produce the hardware design lifecycle data that assures correctness and control of the hardware design lifecycle and its outputs.

- Planning process

The planning process defines "the means by which the functional and airworthiness requirements are converted into a hardware item with an acceptable amount of evidence of assurance that the item will safely perform its intended functions." Four objectives of the planning process are identified, i.e.:

1. Define the hardware design lifecycle processes;
2. Select and define standards against which compliance must be shown;
3. Select or define the hardware development and verification environments;
4. Propose the intended means of compliance of the hardware design assurance objectives with the certification authority.

- Hardware design process

Five major processes which constitute the hardware design process are defined. These processes are not limited to the LRU level only, but can be applied to e.g. circuit board assemblies and programmable logic devices as well. These processes are:

- Requirements capture:
- Conceptual design;
- Detailed design;
- Implementation;
- Production transition.

The hardware design process and its association with the planning and support processes, as described by the standard, is shown diagrammatically in Figure 44.

The RP also affirms that each process, and the interactions between processes, can be iterative. Importantly, for each iteration, the effect of the iterative change on each of the processes should be addressed and evaluated for the impact on the results of previous iterations. This principle forms one of the cornerstones of the process design described in chapter 5.

The RP recognises that current engineering practises provide many different means of representing requirements and design implementations. The guidance provided by the RP is not affected by the selection of a particular representation method. The only restrictions in terms of the design representation method are that the design representation should be sufficient so that the hardware item can be replicated consistently, and that changes in the design representation method should not affect design replication, nor the design assurance level.

The requirements for each hardware item are identified and recorded by the requirements capture process (clause 5.1). The RP recognises the fact that the requirements capture process may be iterative, as additional requirements may be identified during the design process. This iterative process helps to assure consistency of the requirements with the design implementation. Other hardware requirements may also become known during further evolution of system requirements and software requirements.

The following requirements capture activities are summarised from the RP:

- Document system requirements allocated to hardware items;

- Identify safety requirements from PSSA;
- Identify design constraints due to production processes, standards, procedures, technology, design environment and design guidance;
- Determine derived requirements necessary for implementation;
- Indicate derived requirements to system safety process;
- Document requirements data in quantitative terms, with tolerances where applicable;
- Indicate requirement omissions or errors discovered to higher layers of the system hierarchy.

The conceptual design process (clause 5.2) "produces a high level design concept that may be assessed to determine the potential for the resulting design implementation to meet the requirements." Examples of conceptual designs cited in the standard include functional block diagrams, architecture descriptions and chassis sketches.

The primary objective is to develop a conceptual design which is commensurate with the requirements. Derived requirements which are identified while developing the conceptual design; requirement omissions and errors are fed back to the requirements capture process to be recorded or resolved. This loop which exists between the conceptual design and requirements process has important implications in terms of the baselines which are addressed in the process design described in chapter 6.

The following conceptual design activities are summarised from the RP:

- Generate high level description;
- Identify major components;
- Indicate derived requirements necessary for implementation;
- Indicate derived requirements to requirements capture process;
- Document requirements data in quantitative terms, with tolerances where applicable;
- Indicate requirement omissions or errors to appropriate process for resolution;
- Identify reliability, maintenance and test features.

The detailed design process (clause 5.3) produces the detailed design data for the hardware item(s). Derived requirements; requirement omissions and errors in requirements and the conceptual design which are identified during the development

of the detailed design are fed back to the requirements capture process or conceptual design, to be recorded or resolved.

The following detailed design activities are identified:

- Generate the detailed design data in accordance with the hardware requirements and the conceptual design data;
- Implement architectural design techniques as necessary, e.g. safety monitors and fault tolerant design;
- Incorporate test features into the detailed design to allow for the verification of safety features, where necessary;
- Perform an assessment of unused functions and identify potential effects of these on safety, where applicable;
- Constraints on the design, installation or operation of the H/W item that, if not adhered to, could affect the safety of the item, should be identified;
- Indicate derived requirements produced during detailed design process to upstream processes;
- Indicate requirement omissions or errors to appropriate processes for resolution.

The implementation process (clause 5.4) "uses the detailed design data to produce the hardware item that is an input to the testing activity." The implementation process typically produces a "first article" using representative manufacturing processes. Another objective of the implementation process is to ensure that the implementation; assembly and installation data for the hardware item is complete. Derived requirements, requirements omissions and errors should be fed back to upstream processes to be resolved.

The following implementation process activities are identified:

- Produce a hardware item using the design data as well as, where practical, the resources intended for the production product;
- Indicate derived requirements produced during detailed design process to upstream processes;

Indicate requirement omissions or errors to appropriate processes for resolution. The production transition process (clause 5.5) prepares manufacturing data, test facilities and other resources for production. This includes the establishment of a baseline "that

includes all design and manufacturing data needed to support the consistent replication of the hardware item" and the identification and documenting of safety requirements for which manufacturing controls must be established.

The acceptance test (clause 5.6) "demonstrates that the manufactured, modified or repaired product performs in compliance with the key attributes of the unit on which certification is based."

RTCA/DO-254 [26] categorises the validation and verification process, configuration management, process assurance and certification liaison as supporting processes.

- Validation and verification process

The use of the terms validation and verification by RTCA/DO-254 [26] is explicit. It states that "the validation process provides assurance that the hardware item derived requirements are correct and complete with respect to the system requirements allocated to the hardware item. The verification process provides assurance that the hardware implementation meets all of the hardware requirements, including derived requirements.

The RP excludes the validation of hardware requirements allocated from the system requirements as it is assumed that these requirements were validated as part of the system process. The RP also points out that only derived requirements that can affect safety or functional requirements need to be validated.

The RP states that verification consists of reviews, analyses and tests, and these are conducted to provide assurance that the hardware item meets the requirements. As in the instance of RTCA/DO-178C [25], it states that this includes an assessment of the results.

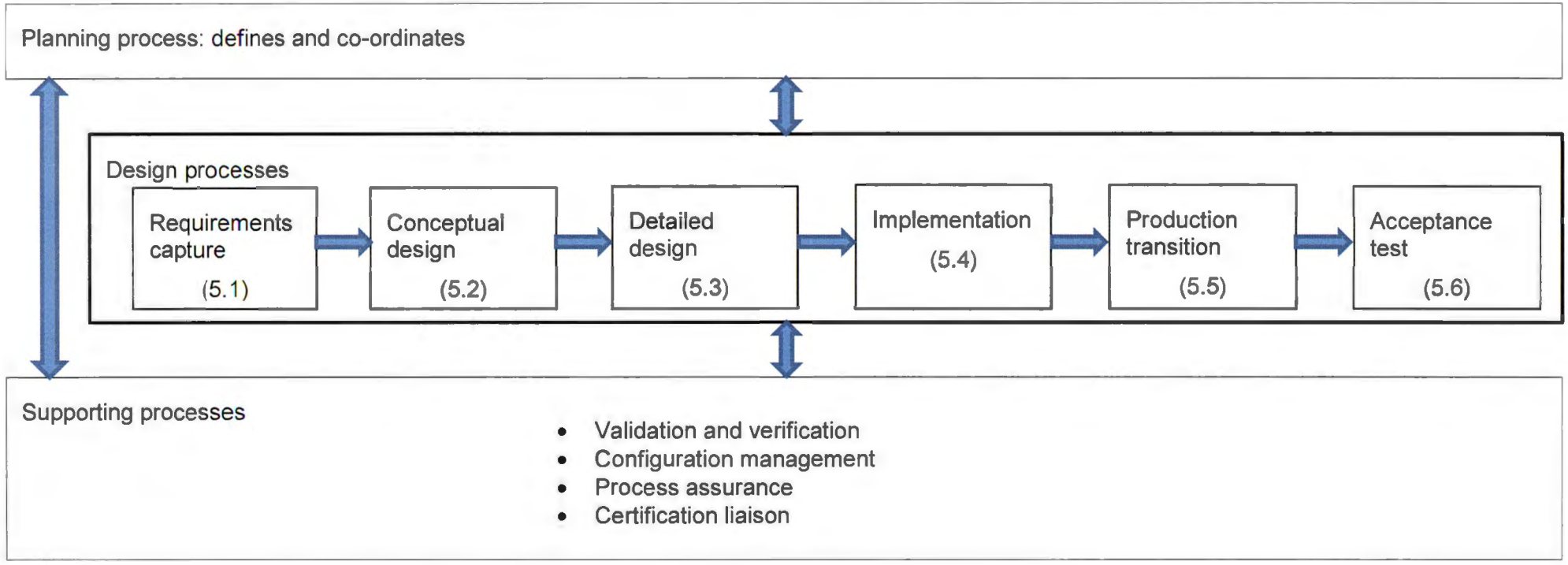

Figure 44: Simplified view of the RTCA/DO-254 process

4.7.6 Development assurance concepts

SAE ARP4754 [33] states that "development assurance is a process involving specific planned and systematic actions that together provide confidence that errors or omissions in requirements or design have been identified and corrected to the degree that the system, as implemented, satisfies applicable certification requirements."

RTCA/DO-254 [26] identifies the term design assurance to encompass "All of those planned and systematic actions used to substantiate, at an adequate level of confidence, that design errors have been identified and corrected such that the hardware satisfies the application certification basis." It describes the objective of process assurance as "to ensure that plans are followed, hardware design lifecycle process objectives are met and activities have been completed.

In more specific terms, these statements can be simplified as follows: For airborne electronic systems, assurance of the integrity of a product is obtained by establishing confidence that errors in the requirements and implementation were detected and removed. This confidence is achieved by providing objective evidence of adherence to a systematic engineering process directed at finding these errors. The rigor by which this process is executed depends on the safety objectives for the product. Documented evidence of adherence to a defined process is used to substantiate claims of airworthiness, i.e. establishing confidence that the developed system is safe for use.

Development assurance levels are assigned as follows: Level "A" processes correspond with functions associated with catastrophic failures, level "B" with hazardous failures, level "C" with major failures, level "D" with minor failures and level "E" when no safety implications were identified. Development assurance for electronic hardware is described in RTCA/DO-254 (the terms development assurance and design assurance are used interchangeably in the standards). The different levels of design assurance are achieved by the levels of thoroughness of the development and verification activities, which are prescribed in the standard.

A very important fact to consider is that software does not fail in the traditional sense, but it contains undetected errors. RTCA/DO-178C [25] defines the software level as "the designation that is assigned to a software component as determined by the system safety process. The software level establishes the rigor necessary to demonstrate compliance with (RTCA/DO-178C)." The processes as directed by RTCA/DO-178B/C aim to provide confidence that errors that may lead to failure events are detected and removed, and it is also achieved by the rigour of the development and verification effort. The software levels are allocated similar to the design assurance levels, e.g. level "A"

software is required for the implementation of functions where failures lead to catastrophic events. The difference between the different software levels is the intensity of verification and validation as well as the degree of independence at which verification and validation tasks are executed. This aspect is discussed in more detail in Chapter 4.7.4.

This can be summarised as follows: for software and "complex" hardware, the confidence that the system is safe for use is established through evidence that the development process supported sufficient measures to ensure that all safety-related errors were detected and removed. This evidence is documented in validation and verification records. Accordingly, the assurance level (development or software) is achieved by virtue of the rigour of the development process, supported by corroborating evidence.

A significant inference from the explanation presented above is that the system safety process relies on the fact that all the system functional requirements, including the derived requirements, were assessed in terms of the hazards associated with failure conditions associated with these functions. The system is developed with the aim of preventing the occurrence of these failures, with a level of effort commensurate with the severity of the failure consequences.

4.7.7 Synopsis of airworthiness recommended practises

4.7.7.1 Scope of engineering activities

The process scope is not explicitly addressed in the airworthiness recommended practises, and is hence not discussed further in this section.

4.7.7.2 Definition of activities and tasks

Systems developed for airborne use must comply with airworthiness requirements, as formulated in the recommended practises reviewed in this study.

Airworthiness processes are based on structured system safety processes. The system safety process relies on a requirements based process, hence the process design presented in Chapter 5 must be a requirements based process.

The recommended practises describe life cycle models appropriate to the development of airborne electronic equipment. The process design presented in Chapter 5 is based on a life cycle model derived from these recommended practises.

Furthermore, the information presented in the airworthiness recommended practises provide useful guidance for the detailed design of process elements.

The elements of a development project, according to ARP4754 [33], which must be incorporated in the development process are re-iterated below.

- Development - the process and methods to be used for the system architecture development, integration and implementation;
- The safety program consists of the safety activities related to the development of the system;
- Requirements management - capturing and management of requirements;
- Validation - showing that requirements and assumptions are complete and correct;
- Implementation verification - processes and criteria which are applied to show that the implementation satisfies its requirements;
- Configuration management - the description of key development related configuration items and the management thereof;
- The process assurance process - ensure that the defined practises and procedures to be applied during system development are followed;
- Certification - the process and methods which are used to achieve certification of the system.

These eight points are fundamental to the development of airborne electronic equipment, hence the process design which is the objective of this study will be structured such that it will show the associations with these points in an unambiguous manner.

The system safety process described by SAE ARP4761 has three main elements, namely a Functional Hazard Assessment (FHA), a Preliminary System Safety Assessment (PSSA) and a final System Safety Assessment (SSA).

The system safety process requires that the functional requirements (including derived requirements) for a system are defined unambiguously and comprehensively. These are assessed in terms of their associated failure conditions, to derive safety objectives for the system. The system design and design processes are influenced by these safety objectives and the final implementation is subjected to verification processes

where compliance with the safety objectives and design and software assurance processes are demonstrated.

Very importantly: for software and "complex" hardware, the confidence that the system is safe for use is established through evidence that the development process supported sufficient measures to ensure that all safety-related errors were detected and removed. This evidence is documented in validation and verification records. Accordingly, the assurance level (development or software) is achieved by virtue of the rigour of the development process, supported by corroborating evidence.

A significant inference from the explanation presented above is that the system safety process relies on the fact that all the system functional requirements, including the derived requirements, were assessed in terms of the hazards associated with failure conditions associated with these functions. The system is developed with the aim of preventing the occurrence of these failures, with a level of effort commensurate with the severity of the failure consequences.

The development process described by RTCA/DO-178B/C constitute the following elements:

- Develop software requirements;
- Develop software architecture;
- Develop software verification cases and procedures;
- Develop software "design" details;
- Implement software "design";
- Integrate software and the hardware elements that constitute the system;
- Verify that executable object code meets with software and system requirements.

A significant consequence of this approach is the life cycle data and artefacts created by this process, directly produces a product that is representative of production versions, i.e. a 1st article.

4.7.7.3 Development process framework

Note that the software development, validation and verification activities as depicted in Figure 43 can be simplified and summarised as shown in Figure 45. The process can be separated into distinguishable groups of activities, i.e.

- Develop software requirements;
- Develop software architecture;
- Develop software verification cases and procedures;
- Develop software "design" details;
- Implement software "design";
- Integrate software and the hardware elements that constitute the system;
- Verify that executable object code meets with software and system requirements.

A significant consequence of this approach is the life cycle data and artefacts created by this process, directly produces a product that is representative of production versions, i.e. a 1st article.

Also note that the requirements process includes identification of derived requirements from downstream activities. It must be ensured that newly identified functional requirements are subjected to an assessment of functional failures and consequences and to ascertain that a verification method or procedure is identified to render verification of the requirement. This principle is fundamental to the airworthiness process prescribed by ARP4754 [33].

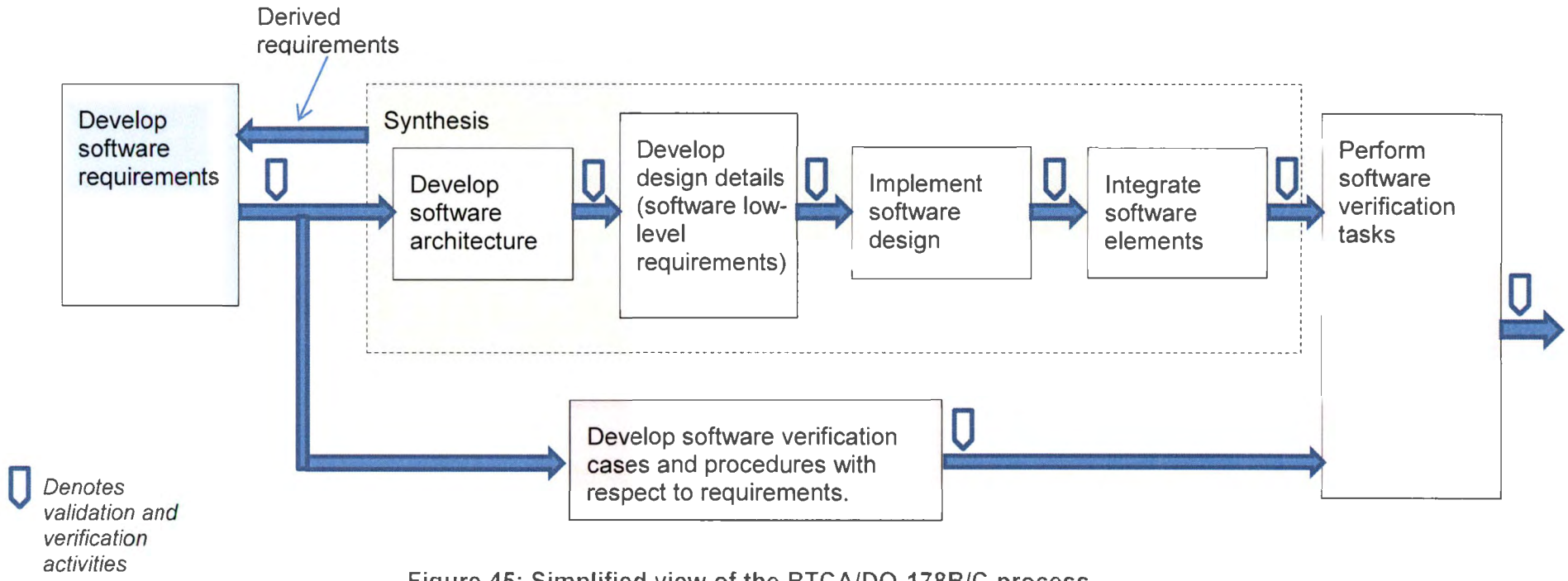

Figure 45: Simplified view of the RTCA/DO-178B/C process

4.7.7.4 Development process controls

The primary objective of the airworthiness recommended practises is to ensure that the system-of-interest, as installed into an airworthy aircraft, does not compromise the safety of the aircraft or its occupants. The assurance that the system is safe is ensured by applying design, process and software assurance levels. These levels prescribe the rigour by which the correctness and completeness of the life cycle data associated with the system is confirmed.

These notions of design, process and software assurance levels underlie the entire airworthiness process and need to be incorporated in the process design.

4.7.8 Airworthiness recommended practises: summary

4.7.8.1 Summary of salient aspects

- The referenced recommended practises establish airworthiness principles to be incorporated in the process design;
- Relies on the concept of development assurance.

4.7.8.2 Summary of concepts applied / discarded

- Invoke system safety processes;
- Specify a requirements based process;
- Describe appropriate lifecycle models.

4.8 Life cycle models described in academic literature

Note that only the third research requirement is addressed in this section.

The topics reviewed in this section are described in academic journals, publications issued by professional organisations e.g. INCOSE, and websites maintained by specific interest groups, e.g. the Body of Knowledge and Curriculum to Advance Systems Engineering (BKCASE). The referenced sources were studied to provide a broader perspective on systems development, as part of the DSR rigour cycle.

4.8.1 Lifecycle approaches

The INCOSE handbook [48] identifies four types of lifecycle approaches, i.e. plan-driven methods, incremental and iterative development (IID), lean development, and agile development. These are summarised here to provide a more complete background to the problem of selecting a suitable approach to the development of a dedicated process.

4.8.1.1 Plan-driven methods

The plan driven approach is recommended for projects where the work performed by large teams of people needs to be co-ordinated. The following quote from the handbook clarifies the concept:

> "Plan driven methods are characterised by a systematic approach that adheres to specified processes as the system moves through a series of representations from requirements through design to finished product. Specific attention is given to the completeness of documentation, traceability from requirements, and verification of each requirement after the fact."

It is further pointed out that plan-driven methods provide predictability, stability, repeatability and high assurance. Importantly, it is emphasised that safety-critical products can only meet modern certification standards by following a thorough, documented set of plans and specifications.

4.8.1.2 Incremental and iterative development

Plan driven methods are practical when requirements are defined to a sufficient level at the outset of a project, or when the plan incorporates a prototyping stage to support the requirements development process. In instances where series production of a system is not anticipated, e.g. a computerised system for managing a specific task at an enterprise where the product will not be replicated, it is conceivable that the requirements for the system are not all known at the outset of the project. A form of incremental and iterative development (IID) (not be confused with incremental and iterative development where this forms part of a structured plan, as described in Section 4.8.3.3) is an approach to develop the system requirements by developing small increments which are then evaluated and elaborated.

An important aspect of IID is highlighted in the INCOSE handbook [48]. IID is distinguished from the plan-driven way of doing things in terms of velocity and

adaptability. By changing velocity, it means that the speed as well as the direction in which a project advances are changed, in response to feedback from the customer or project sponsor. A danger exists in this method in that frequent changes in direction can have chaotic consequences.

4.8.1.3 Lean development

Lean development has as its aim the elimination of wastage due to, amongst other, co-ordination problems, unstable requirements and quality problems. In short, lean methods aim at the reduction of over-processing, waiting time, unnecessary movement, over production, transportation, inventory and defects. Lean methods address a wider scope of a system's lifecycle than the segment of the lifecycle which is the focus of this study. These methods do not readily apply to the problem at hand and are therefore not discussed further here.

4.8.1.4 Agile development

The INCOSE handbook [48] juxtaposes agile development with the orderly, hierarchical baseline progression followed by a verification sequence promoted by conventional methods.

The "Manifesto for Agile Software Development" [57] was published in 2001, and was an attempt to improve the quality and speed of delivery of software in an environment where customer satisfaction is not regularly achieved.

The manifesto is a useful counterpoint for the assessment of a formalised development process. An interpretation of "agile" principles with respect to structured development is presented below.

"Individuals and interactions over processes and tools"

In reality work outcomes are achieved by a combination of the two. Processes and tools should be designed to support individuals and interactions, and not the other way around. However, in a project where evidence of process compliance is used in order to obtain airworthiness approval, the process aspect cannot be ignored.

"Working software over comprehensive documentation"

Documentation should add value; otherwise preparing documentation is a waste of time. And approved documentation without working software is meaningless. MIL-STD-498 [10] is an example of a documentation driven process, and inefficiencies in this process has already compelled the DoD to cancel it. However, proper

documentation mitigates risks of miscommunication or loss of critical personnel. It also supports long term maintenance and upgrading of the software, and is required in providing evidence of process compliance.

"Customer collaboration over contract negotiation"

This argument comes about due to the problem of identifying user requirements sufficiently at the outset of the project. But this means that in a formal contracting environment sufficient effort must be allowed for the determination of the system requirements. If there are uncertainties at the time of contract negotiation, provision must be made for updates to the contract when these are resolved. This element of the manifesto increases the risk for the customer and decreases it for the enterprise; it will only be acceptable in cases where there is an exceptional working relationship and trust between the two parties.

"Responding to change over following a plan"

A plan should not be followed if it does not make sense. The purpose of a plan is to communicate strategies and intent to all involved parties. If these need to change, the plan should change as well. The challenge is to write the plan such that it allows sufficient flexibility in low level decision making while still ensuring coherence at high level.

In summary, the "Agile", as well as "skunk works" approaches aim to resolve the issue of starting off from a comprehensive set of requirements. These all have merit, provided the funding model for the project is compatible with the approach.

4.8.2 Types of lifecycle models

A lifecycle model can be considered to be a presentation of the steps by which the engineering process proceeds through the system's lifecycle.

In the Guide to the Systems Engineering Body of Knowledge (SEBoK), as published in BKCSE [59] it is pointed out that the selection of a lifecycle model depends on the specific situation where the model is to be applied, i.e. there is no "one-size-fits-all" model that fits every situation. Reference is made to two classes of lifecycle models, i.e. the pre-specified, sequential single-step model, and incremental and evolutionary models. The different models, their merits as well as disadvantages in different situations, are described below.

An observation that must be made at this point is that the concept of a lifecycle stage only has significance if traceability between the lifecycle data produced during different

stages is to be shown, and if the lifecycle data associated with a particular stage is subjected to some measure of control, e.g. requirements being signed off prior to implementation. If changes can be made to lifecycle data and artefacts without controls, and traceability is not important, then there is no point to discerning specific lifecycle stages.

4.8.2.1 Lifecycle model selection criteria

The BKCSE [59] identifies three stages which are common to all the lifecycle models, i.e. a definition stage, a development stage, and the production, support and utilisation stage.

It also presents a decision table to provide guidance in terms of the model to be selected for a specific application, based on the questions:

- Is the project characterised by stable, pre-specifiable requirements?
- Is it acceptable to wait for development to be completed before starting to utilise the system?
- Is there a need to get operational feedback from an initial capability before developing further increments?
- Is there a need to wait for particular enablers, e.g. technology maturity, external system capability or needed resources before the next developments stage can commence?

The selection of an appropriate lifecycle model can be rationalised by answering these questions with reference to a specific situation.

4.8.2.2 Lifecycle model classes

- Pre-specified, single-step

The BKCSE [59] refers to the pre-specified, sequential single-step model as the traditional lifecycle model. This model is shown schematically in Figure 46.

Figure 46: Pre-specified, sequential single-step lifecycle model

In this model, system or product specifications are developed during the definition stage, and it is assumed that these specifications remain stable throughout the total lifecycle of the system or product. This model is applicable to simple manufactured products, such as bolts and nuts. The BKCSE [59] states that an advantage of this method is that in appropriate situations it is efficient and easy to verify, but drawbacks are that it is difficult to implement in situations where rapid changes or emerging requirements are to be managed, such as in human intensive systems. If the answer to the first two questions stated above is "yes", then this lifecycle model is appropriate. However, comparison of Figure 46 with Figure 14, which represents the classical systems engineering approach, shows that, in principle at least, the classical method could also be classified as having employed a pre-specified, sequential single step lifecycle model, which was applied in the development of complex systems. Methods to resolve the difficulties identified in the BKCSE [59] when applying the pre-specified, sequential single step lifecycle model in the classical method were discussed in Chapter 4.3.3.3. It was shown that the refinement of requirements in the single step approach was addressed through the use of a number of prototype models.

All other lifecycle models identified by the BKCSE [59] describe some form of iterative or incremental development. These are considered below.

- Pre-specified, multi-step

In a pre-specified, multi-step project the development block is split into more than one increment, in order to provide an early initial (limited) operational capability, followed by pre-planned product improvements. An alternate version of this model performs the development in blocks, but do not deliver the product until completion of the final development block. These views are shown in Figure 47 and Figure 49.

An advantage of this model, compared with the single-step model, is that it provides an early initial capability, and as with the single step model, it enables a strong, predictable process. The drawbacks are the same as for the single-step model, i.e. it is difficult to implement in situations where rapid changes or emerging requirements are anticipated. When the answer to the first question is "yes" but to the second is "no", then this process should be implemented on the project.

Figure 47: Pre-specified, multi-step lifecycle model

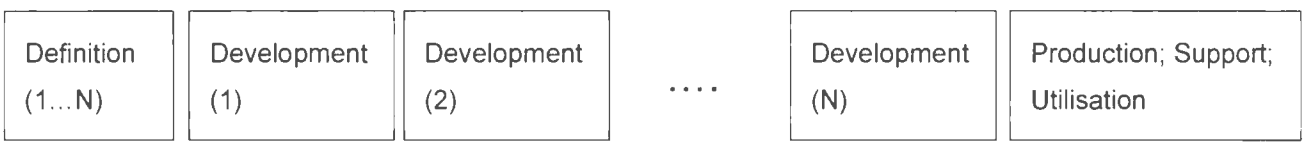

Figure 48: Pre-specified, multi-step lifecycle model, alternate version

- Evolutionary, sequential

The evolutionary, sequential lifecycle model represents a situation where the initial capability of the system is developed and fielded rapidly and updated regularly, where the updates are based on operational experience. The first increment is fielded and utilised while the definition and development of the second increment is undertaken, and so forth, as shown in Figure 49. An obvious advantage of this model is the adaptability to change. A disadvantage pointed out by BKCSE [59] is that it can inadvertently encourage a situation where the easiest set of architectural commitments are implemented first, leading to difficulties if the workload needs to be scaled up when more complex functionality must be implemented.

When the answer to the first two questions stated in Section 4.8.2.1 is "no", but the answer to the third question is "yes", this model should be selected for the project.

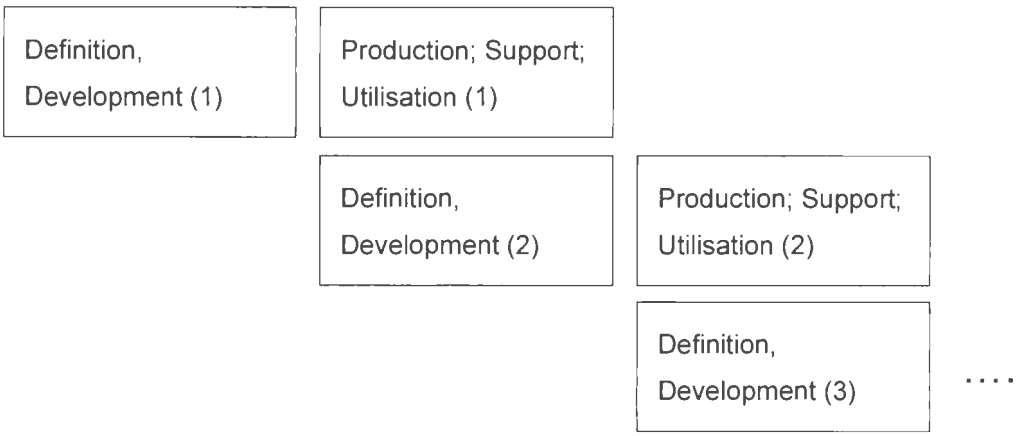

Figure 49: Evolutionary, sequential lifecycle model

- Evolutionary, opportunistic

The evolutionary, opportunistic lifecycle model describes a similar concept as the evolutionary sequential model, but the definition and development stage of a subsequent increment is not associated with the utilisation of the preceding increment. This model is shown in Figure 50 and is typically applied when the decision to commence with the next definition and development stage is dependent on factors other than emerging new requirements, e.g. maturing of an appropriate technology or availability of resources. Another instance is when an external system which is necessary for the operation of the system becomes operational. This model should be selected if the answers to the first three questions as posed in Section 4.8.2.1 are "no" but the answer to the last question is yes.

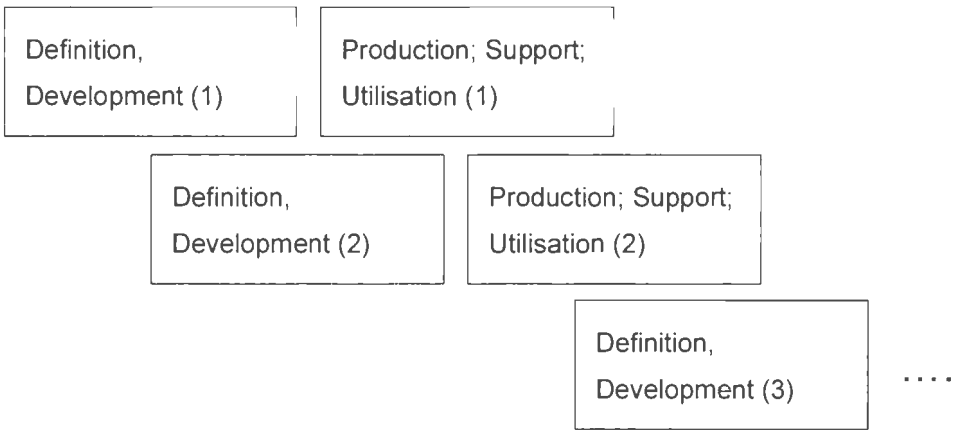

Figure 50: Evolutionary, opportunistic lifecycle model

- Evolutionary, concurrent

The evolutionary, concurrent model applies to the situation when the answer to all the questions posed in Chapter 4.8.2.1 is "no". In this model, an initial development increment is associated with an initially specified set of requirements. A second, related

cycle of definition and development, takes place alongside this initial cycle, and can be followed by more similar cycles, with limited dependencies between the cycles. This model is typically selected when different engineering teams develop functionality concurrently.

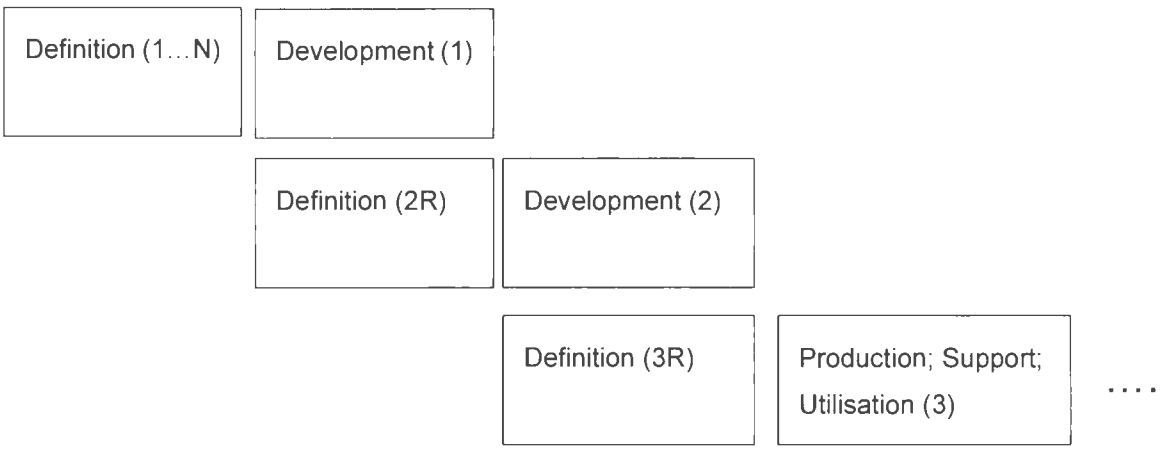

Figure 51: Evolutionary, concurrent lifecycle model

4.8.3 Review of other published lifecycle models

A number of software development lifecycle models depicted in academic literature are described below. The significance of these models with reference to the development of airborne electronic systems is discussed, and the models are compared with the classifications described in the preceding section.

4.8.3.1 Chaotic approach

Some projects can be classified as "chaotic", i.e. the entire project is executed on an entirely ad-hoc basis. Siddiqui et al [60] pointed out that the success of failure of a project which employs no structured process at all is solely dependent of the capability of the individuals involved and that such an approach may be appropriate to small projects, but is inadequate for medium and large projects where discipline and coordination become necessary to achieve set goals.

The chaotic approach cannot be classified under one of the lifecycle model types described below.

4.8.3.2 Waterfall model

The term "waterfall model" was used in a paper published in 1976 by T. E. Bell and T. A. Thayer [61], to characterise the top-down approach for the development of software,

in this instance for application in a ballistic missile defence system. The paper cited MIL-STD 490/483 which specified a systems requirements document, a "design-to" requirements document created in response to the system requirements, and then a "code-to" requirements document for each software module in the design. In this progression each new document is at a more intensive level of detail, i.e. a top-down process. Bell and Thayer in turn cite a 1970 paper by Winston W. Royce [62] who indicated the logical progression of development of a software based system, starting from the system requirements, through software requirements, analysis, program design, coding, testing and operations. In this view, each step is to be planned and staffed differently for best utilisation of resources. Figure 2 from Dr Royce's paper is redrawn below in the same format as presented in his paper, and the "top-down" workflow can be seen to emulate a cascading waterfall, hence the origin of the term.

This "waterfall" description of the development process provides a clear-cut graphical view of the essence of a development project. However, it is interesting to note that Royce [62] already identified shortcomings of this pure sequential approach to development. He points out that iterations invariably occur between successive steps in the sequence, where improved understanding of the details at a certain step requires updates to the preceding step. As long as these iterations exist between adjacent steps only, the changes can be managed effectively. However, changes are not limited to this simple scenario.

For example, a timing problem discovered during testing may require updates to the specifications, which then has a significant impact on all subsequent steps. The objective of Royce's paper was to discuss possible solutions to this problem of managing design information feedback. In essence, in this paper it is suggested that the entire development cycle is executed twice. Siddiqui et al [60] indicate that this model still remains relatively popular but mention the fact that software intensive projects rarely adhere to this model, and that it does not allow feedback loops. Once requirements are written they are assumed to be stable throughout the project and the process makes no provision for revising the requirements.

It must be noted that conceptually, the classical "military" lifecycle described in Section 4.3.1 resembles a waterfall model. Due to the single progression through the steps, the waterfall model resorts under the class of pre-specified, single step models described in Chapter 4.8.2.2.

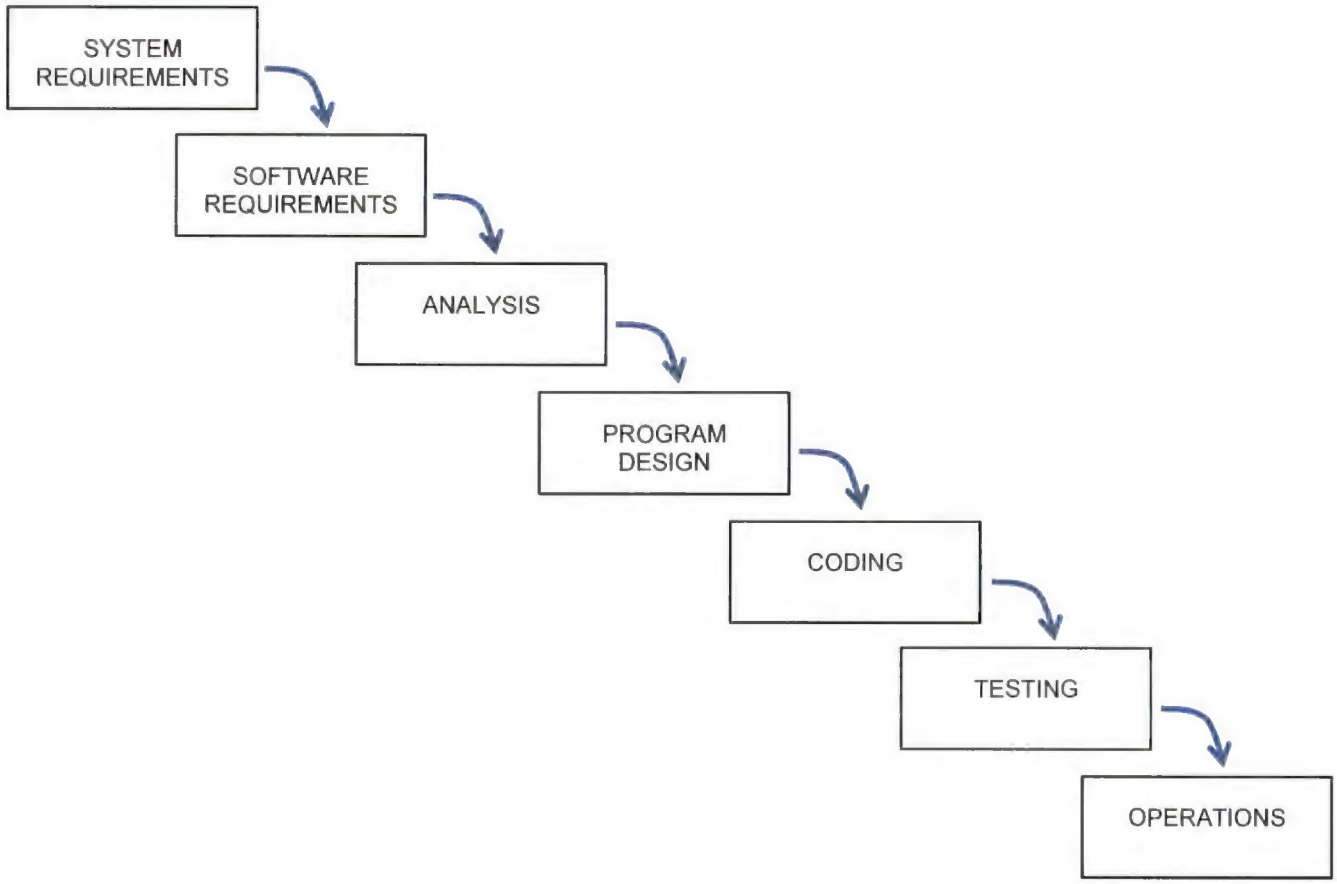

Figure 52: Implementation steps to develop a large computer program for delivery to a customer (Royce [62])

4.8.3.3 Structured iterative development

Siddiqui et al [60] identifies an iterative model in which the project is fragmented into smaller components, each representing the waterfall model described above. In an iteratively developed project, each increment allows for refinement of requirements, design and code. This methodology reduces the risk of uncovering significant problems during advanced stages of the development cycle, as pointed out above.

This approach increases the likelihood that the client will increase demands in functionality as the project progresses (scope creep). This aspect, as well as the number of versions of lifecycle data produced, require an efficient change control mechanism.

Another version of structured iterative development is a model where requirements are firmly established, and the development of functionality and delivery to the user is fragmented into blocks. This approach eliminates the scope creep problem, but also

relies on effective change control methods. It also speeds up the delivery of (at least) partial utilisation of the system.

This highly structured incremental model is an example of the evolutionary, sequential model described in Chapter 4.8.2.2.

In a 2003 paper by Larman and Basili [58], the advantages of an incremental and iterative approach are advocated. They point out that IID has its origins in the mid-1950s. The development of Project Mercury, the first human spaceflight programme undertaken by the United Sates which commenced in 1958, is cited as an example of successful application of IID to software development. The advantages of incremental development, and the widespread use of the concept, is described in this paper. Importantly, a conclusion from this paper is that most projects follow some form of IID.

4.8.3.4 Spiral model

The spiral model for software development was introduced by Barry Boehm in 1986, and the concept was further expounded in subsequent publications, notably in a CMU/SEI special report [63]. The model is characterised by repeatedly iterating a set of elemental development processes in a series of phases. In each iteration, risks are identified and resolved. The process promoted the development of a series of prototypes which were used to improve understanding of the system to be realised, and these supported planning of subsequent phases, and analysing of risks for each stage. In the first loop of the spiral, a concept of operation is developed, in the next loop, software requirements are developed and validated, in the subsequent loop the software product is designed, validated and verified, and in the final loop, a detailed design is produced, source code is developed and subjected to unit testing, the system is integrated onto the target platform and subjected to acceptance tests, and then finally implemented.

Criticism levelled at this model by Siddiqui et al [60] is that delays in production due to the many steps involved can be serious. The model is also not generally applicable and they suggest that this model is only applicable to projects which involve major risks.

Although the spiral model describes a number of iterations, the progression from definition through development to production, support and utilisation is linear and involves only a single delivery increment. Hence this model is also classed under pre-specified, single step models as described in Chapter 4.8.2.2.

4.8.3.5 Prototyping model

Siddiqui et al [60] identifies a lifecycle model based on prototyping. A simplified version of the intended system is built for evaluation purposes, and feedback from stakeholders is used to produce a second, improved version for further evaluation, and this process is repeated until the full set of requirements is established.

For projects where requirements are not well understood at the outset, this approach is appropriate in terms of the definition stage. A risk associated with this approach is that false expectations can be rendered with the customer, who may be under the impression that the work remaining to produce the final system is less than what is actually required.

Note that there are similarities between this model and the spiral model described in Chapter 4.8.3.4. A process where prototypes are developed to establish the system requirements, i.e. to complete the definition stage, followed by a development stage, and then delivered, is also an example of a pre-specified, single step model as described in Chapter 4.8.2.2 as development only commences after completion of an (elaborate) definition stage.

4.8.3.6 Re-use model

In this model, a system is largely built from existing components. In the case of software, this relates to object-oriented development. This model is useful if a repository of software modules which can be utilised on a specific project, is readily available. Depending on the specific project none of the models described in Section 4.8.2 are applicable, as the development stages are insignificant (no development required, components are re-used).

4.8.4 Lifecycle models described in academic literature: conclusions

As pointed out in in the introduction, only the third research challenge, i.e. the determination of a lifecycle model appropriate for the development of airborne electronic equipment, was addressed in this chapter.

The INCOSE handbook [48] identifies four types of lifecycle approaches, i.e. plan-driven methods, incremental and iterative (IID) development, lean development, and agile development. The BKCSE [59] presents a number of lifecycle models, but differentiates between pre-specified and evolutionary classes of models.

Examination of these shows that there are essentially only two fundamental classes of approaches, i.e. planned and structured development, and less structured

development. As pointed out in the INCOSE handbook, a planned approach is characterised by adherence to specified processes which direct the development effort from requirements through design to finished product, and attention is given to the completeness of documentation, traceability from requirements, and verification of compliance with requirements. It was shown in the preceding chapters in this literature study that this approach is mandatory when developing airborne electronic systems.

Therefore, as pointed out in previous sections of this dissertation, the plan driven approach is to be adopted for the development of airborne electronic equipment.

There are a number of lifecycle models which can be implemented in a planned and structured approach. It was shown that the classical military development lifecycle could be described as having employed a pre-specified, sequential single step lifecycle model, supported by the use of prototypes.

The incremental implementation of functionality is described by the BKCSE [59] as a "pre-specified, multi-step" model, and two versions of the model are presented, i.e. the delivery of incremental blocks of functionality, or to execute the development in "blocks", but to deliver the system after integration of the final block into the system. Both these versions can be considered in the development of airborne electronic equipment, depending on the operational requirements of the user.

Consequently, in terms of the classifications described by the BKCSE [59], the *lifecycle model appropriate to the development of airborne electronic equipment is the pre-specified, multi-step model.*

However, the less structured development approaches warrant further comment, as a study of attributes of these approaches may improve the understanding of development processes. It is important to recognise that a number of challenges concerning the development of complex systems, e.g. the curbing of time and budget overruns, are due to the difficulty to establish a comprehensive set of requirements at the outset of a project. Another major problem encountered in the development of complex systems is the difficulty to estimate the scope of work accurately, at the beginning of the project. These concerns give rise to the many different lifecycle types presented in the literature, all of which endeavour to resolve these problems.

Agile development promotes a flexible approach where requirements evolve over time and the development process as well as the agreement adapts to changes which result from new or altered requirements. This approach can best be described as an example of an evolutionary, opportunistic lifecycle model, according to the definition presented by the BKCSE [59].

Due to development process constraints emanating from e.g. the recommended airworthiness practises, e.g. design and process assurance requirements, agile methods are not suitable for the development of airborne electronic equipment.

4.8.5 Lifecycle models described in academic literature: summary

4.8.5.1 Summary of salient aspects

- Different lifecycle model types are described, appropriate to different scenarios;
- The lifecycle models presented in academic literature support the notion of difficulty to establish requirements up front;
- Support notion of difficulty to estimate scope of work accurately.

4.8.5.2 Summary of concepts applied / discarded

Apply:

- Pre-specified, multi-step model;
- Iterative and incremental development.

Discard:

- Agile methods.

4.9 Process synthesis based on the literature study

4.9.1 Scope of engineering activities

The context of the processes for engineering of systems in the wider environment where these processes are executed, according to EIA-632 [12], was briefly described in Section 4.4.1 and is graphically presented in Figure 25. The standard places the processes for engineering of systems within a project environment that exists in an enterprise environment. The enterprise operates in the larger external environment. These contextual aspects are described below, as they apply to the development of airborne electronic equipment.

4.9.1.1 Technical processes

- External environment

The organisation or enterprise responsible for the execution of a project is subject to external influences that determine how the organisation conducts its business.

A list of such external influences, as summarised from IEEE STD 1220 [38] and EIA-632 [12], is presented in Table 11.

Influence	Reference
Laws and regulations	EIA-632; IEEE 1220
Legal liabilities	EIA-632
Social responsibilities	EIA-632
Technology base	EIA-632; IEEE 1220
Labour pool	EIA-632
Competing products	EIA-632; IEEE 1220
Standards and specifications	EIA-632; IEEE 1220
Public culture	EIA-632

Table 11: List of External Influences

In terms of laws and regulations, social responsibilities, labour pool and public culture it can be generally assumed that enterprise wide processes are sufficient to address these influences, and that these aspects do not influence the design of the engineering process.

In terms of legal liabilities however, it is important to realise that all outputs of the development processes must be of such a standard that it will stand up to scrutiny in a court of law, as described in Chapter 3.3.3. Therefore, the development process must be structured in such a manner that its outputs will meet with this objective.

When embarking on a new development, careful considerations should be given to the selection of the technology base, which is intended to be implemented in the product and its enabling products, to ensure that the selected technologies are sufficiently mature as to reduce implementation uncertainties, but not at risk of becoming obsolete. Suitable measures should exist in the development process to ensure that this issue is addressed.

As the products described in this study are all developed in response to a higher level requirement, market environment aspects, such as competing products, are not considered further.

Aircraft operate in the public domain and are therefore subject to certification against regulations and standards as determined by airworthiness authorities.

- Enterprise environment

Enterprise environment influences, as summarised from IEEE STD 1220 [38] and EIA-632 [12], are presented in Table 11.

Influence	Reference
Policies and procedures	EIA-632; IEEE 1220
Standards and specifications	EIA-632; IEEE 1220
Guidelines	EIA-632; IEEE 1220
Domain technologies	EIA-632; IEEE 1220
Resources	IEEE 1200
Local culture	EIA-632

Table 12: List of Influences Internal to the Enterprise

The enterprise normally controls a set of directives by which it manages the ways it conducts its business. These directives vary from mandatory instructions to general guidelines. These directives generally directly influence the project planning processes. Other aspects, internal to the enterprise, that influence the development process are the capabilities accrued by the different departments in the organisation.

Policies and procedures capture the principles and methodology by which particular tasks are executed in the enterprise. These form the core of the corporative quality management system, and are also discussed in more detail in section 4.5.1.

Internal standards and specifications are generated, against which the outputs of development activities are measured, e.g. design and coding standards. These standards can be enterprise wide or can be developed for use in a specific project.

When instructions on how to execute particular development activities are generated to direct team members, but the instructions are not mandatory, the instructions are published as guidelines. These guidelines can be very useful as they record the accumulated experience of the development team. If they are well documented, they

also provide an efficient mechanism for continuous process improvement, as procedural errors can be pinpointed and the documentation can be updated to eliminate errors and inefficiencies as they are detected.

The enterprise selects domain technologies appropriate to its business and areas of expertise. The acquisition and mastery of a particular technology can be costly. Equally important is the loss of a capability associated with an acquired technology. It logically follows that the decision making processes around technology acquisition and retention must be managed with diligence.

The cost of a development program is significantly affected by the acquisition of resources required for the execution of the program. On the other hand, a competitive edge can be established if the enterprise possesses these resources. The development, retention and disposal of resources related to the development process needs to form an integral part of the development support framework.

The local culture within an organisation can influence the level of control required. Although cognisance must be taken of this fact when executing a development process/project, it is not discussed in further detail in this study, as human behaviour was explicitly excluded.

- Project environment

EIA-632 [12] divides the project environment into the "process groups for engineering systems" and the project support functions, which include project management and agreement support.

Work activities executed at the project level are tailored to meet the requirements of the project. The project details are communicated via directives, procedures and plans. The primary objective of these is to provide guidance to the development team, and to provide a common frame of reference of product wide technical decisions to all stakeholders.

Development progress is assessed through project reviews, metrics and controls. This assessment is performed against published directives, procedures and plans.

The separation between project management and technical processes, for the purpose of deriving a life cycle model for development of airborne electronic equipment, is described next.

The project management function performs the liaison between the enterprise, customer, and technical processes. ISO/IEC15288:2008 [39] (Clause 6.3.2) states that the project assessment and control activities "determine the status of the project

and direct project plan execution to ensure that the project performs according to plans and schedules, within projected budgets, to satisfy technical objectives".

In this context, project management activities include the following:

- Management of the agreement;
- Planning and management of the schedule – delivery and events; and
- Planning and management of allocation of resources.

ISO/IEC15288:2008 [39] (Clause 6.4) identifies the technical processes as those processes that:

> "[Are] used to define the requirements for a system, to transform the requirements into an effective product, to permit consistent reproduction of the product where necessary, to use the product to provide the required services, to sustain the provision of those services and to dispose of the product when it is retired from service."

The technical processes produce a qualified representation of the system-of-interest and associated life cycle data.

The technical processes that must be included in the new process description typically encompass the following activities:

- Technical planning and management;
- Requirements management;
- System design;
- Prototyping and modelling;
- Manufacture of hardware
- Software coding;
- Integration;
- Verification and validation.

These activities also include speciality areas such as:

- Airworthiness;
- System safety;
- Reliability;

- Security;
- Maintainability;
- Supportability;
- Human factors;
- Electromagnetic compatibility;
- Other subject matter expertise.

The detailed specification of the process scope and context, based on this research result, is presented in Section 1.1.

The scope of the process for which a design is to be developed is now clearly established in terms of process boundaries and interfaces, and hence provides a partial solution to the first research challenge.

4.9.1.2 System decomposition considerations

SAE ARP4754 [33] defines complexity as "an attribute of systems or items which makes their operation difficult to comprehend." It furthers states that an "[increase] in system complexity is often caused by such items as sophisticated components and multiple relationships".

RTCA/DO-254 [26] states:

> "A hardware item is identified as simple only if a comprehensive combination of deterministic tests and analyses appropriate to the design assurance level can ensure correct functional performance under all foreseeable operating conditions with no anomalous behaviour. When an item cannot be classified as simple, it should be classified as complex. Items that contain a device such as an ASIC or PLD can be considered simple if they meet the criteria of simple as described (here)."

Only systems consisting of a few elements and involving a small number of individuals can be created without some or other type of formal organisation. In contrast, an organised system with many interacting elements is complex by definition. The systems under investigation in this study can generally be viewed to be complex.

Furthermore, due to the complexity of the system to be developed, it can be inferred that the process to build and qualify the system is also complex by nature. A logical step in terms of dealing with a complex system is to decompose the system into

subordinate constituents. There are discernible differences between the system break-down approach stipulated by classical systems engineering principles and the contemporary view. The contemporary view on system decomposition provides a significantly simplified interpretation.

A similar view in terms of systems decomposition is described in Clause 5.1 of ISO/IEC 15288-2008 [39]. It states that:

> "The perception and definition of a particular system, its architecture and its system elements depend on an observer's interests and responsibilities. One person's system-of interest can be viewed as a system element in another person's system-of-interest."

It identifies the following key points regarding the characteristics of the system-of-interest, which apply to the type of system that forms the subject for this study:

a) Defined boundaries encapsulate meaningful needs and practical solutions;

b) There is hierarchical or other relationship between system elements;

c) An entity at any level in the system-of-interest can be viewed as a system;

d) A system comprises an integrated, defined set of subordinate system elements;

e) Characteristic properties at a system's boundary arise from the interactions among system elements.

In terms of the system structure it defines a system to be "composed of a set of interacting system elements, each of which can be implemented to fulfil its respective specified requirements".

It the follows that, for a system that typically consists of electronic hardware hosting embedded software and installed in an airworthy enclosure, the practise as portrayed in Figure 27 is appropriate, and that the system hierarchy can generically be broken down, as presented in Figure 53.

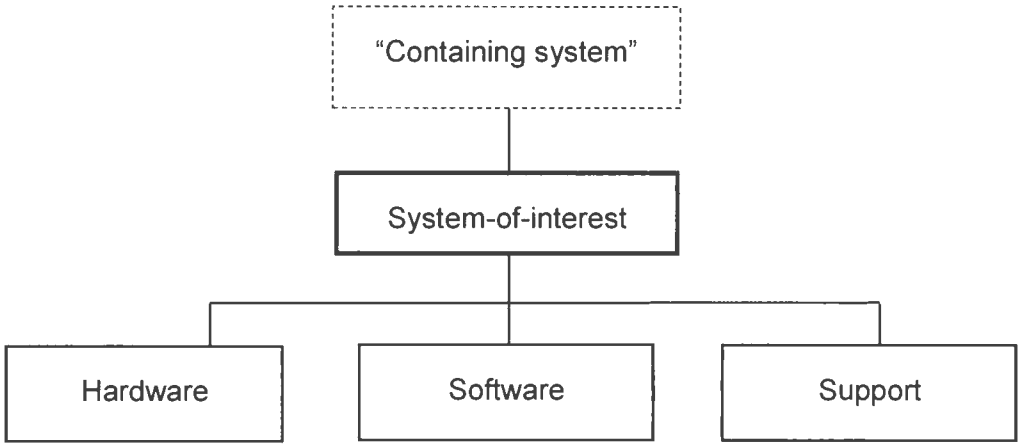

Figure 53: Generic system hierarchy

This view can be corroborated by a review of the projects that form the case studies associated with this research.

The Rooivalk AFCS (which was a subsystem of the helicopter, which again formed part of a larger operational system) was the containing system, the AFCS computer was the system-of-interest, the enclosure and electronic cards formed the hardware, the embedded software as well as the software development environment formed the software element, and the maintenance support test systems, the operating and maintenance procedure manuals formed the support system. A breakdown of the Rooivalk AFCS is shown in Figure 54.

The Oryx navigation system could be broken down in a similar decomposition, as shown in Figure 55, representing the Avionics Interface Unit as the system-of-interest.

From the preceding discussion it follows that the development process design, which is the desired outcome of this study, should accommodate the system breakdown structure as presented in Figure 53.

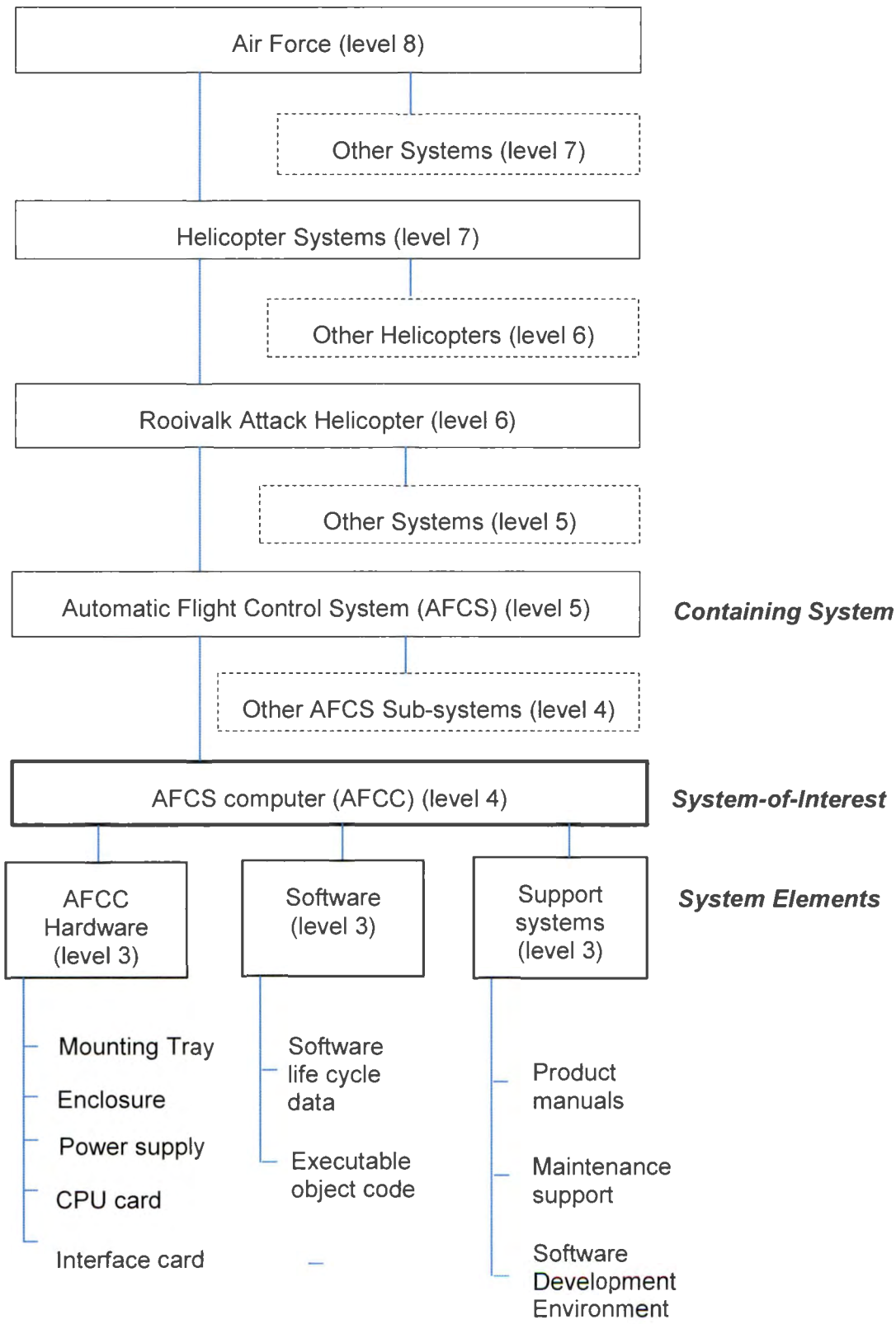

Figure 54: Rooivalk AFCS computer system hierarchy

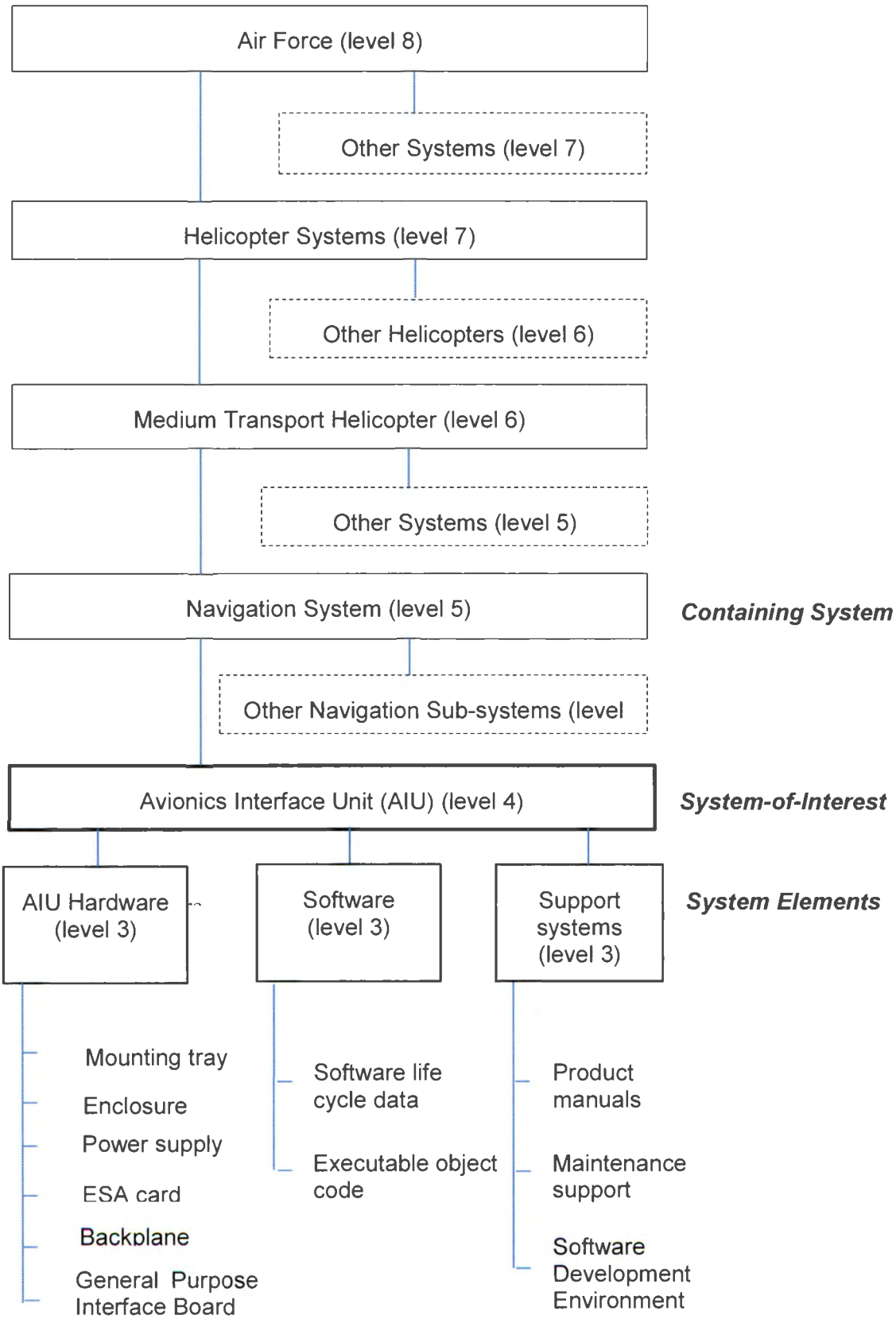

Figure 55: Oryx Avionics Interface Unit (AIU) hierarchy

By employing the system of systems concept, the method for describing the process at different layers of hierarchy is greatly simplified. This affects the scope of the development process as similar process activates are used generically at all layers of the system hierarchy.

4.9.2 Definition of activities and tasks

The second research challenge stated that the activities and tasks necessary for the development of airworthy electronic systems are to be identified.

In order to obtain a framework for the examination of these activities, a method for organising the lifecycle processes into manageable groups is derived in the following section.

4.9.2.1 Requirements processes

The requirements process establishes all the requirements that the system-of-interest needs to comply with. The formulation of the requirements is important for a number of reasons. It is evident that the requirements specify the desired properties of the system to be developed, but the requirements also provide a clear basis for the qualification and certification of the system, and the system safety process used by the airworthiness community is based on a sound requirements management process.

The following concepts and terms apply to the requirements process.

- Requirement

A requirement is defined in SAE ARP4754A [33] as "an identifiable element of a function specification that can be validated and against which an implementation can be verified".

RTCA/DO-178C [25] identifies "high level" and "low level" requirements, specifically with reference to software development. It identifies high level requirements to be "software requirements developed from analysis of system requirements, safety-related requirements, and system architecture". These typically specify functional and performance requirements. Low level requirements are "software requirements derived from high level requirements, derived requirements, and design constraints from which source code can directly be implemented without further information." In most instances, low level requirements can be construed to imply a software design set.

Note that the topic of requirement characteristics and requirement types is a comprehensive subject on its own and not within the scope of this study.

- Derived requirement

According to e.g. SAE ARP4754 [33], derived requirements are "additional requirements resulting from design or implementation decisions during the development process. Derived requirements are not directly traceable to higher-level requirements, though derived requirements can influence higher-level requirements." Examples of typical derived requirements, as they apply to system implementation are provided in e.g. RTCA/DO-254 [26] Section 5.1.2. (4).

The identification of derived requirements is an important aspect of the system safety process described in SAE ARP4761 [34], as derived requirements must also be subjected to the requirements based safety assessments processes.

As no traceability to higher layer requirements can be demonstrated for a derived requirement, it is imperative that the evidence for the existence of derived requirements is provided during the requirements management process (see requirements validation below).

- Specification

For the purposes of this dissertation, the definition of a specification, as defined in RTCA/DO-254 [26], is appropriate, i.e. a specification is "a collection of requirements that, when taken together, constitute the criteria which define the functions and attributes of an item". Key to this definition is the notion "collection of requirements". It must be understood that a specification does not necessarily have to be in the form of a document, but that information in a database can also be viewed as a specification, provided sufficient configuration management is enforced.

- Constraint

In terms of the development of a system, IEEE STD 1220-2005 [38] defines a constraint as "a limitation or implied requirement that constrains the design solution or implementation of the systems engineering process and is not changeable by the enterprise". This notion can be extended to be applicable to the design of a process as well. ANSI/EIA-632 [12] defines a constraint as "(1) A restriction, limit, or regulation imposed on product, project or process; (2) A type of requirement or design feature that cannot be traded off".

- Requirements validation

All the references reviewed in this chapter require that the requirements are validated prior to implementation. SAE ARP4754 [33] defines validation as "the determination that the requirements for a product are sufficiently correct and complete".

4.9.2.2 High level design

The references studied in this chapter commence the synthesis process with some or other form of high level design. ANSI/EIA-632 [12] describes the development of a "logical solution and a physical solution" (Requirements 17 and 18); ISO/IEC 15288-2008 [39] identifies an "architectural design process" (Sub-clause 6.4.3); the process description in IEEE STD 1220-2005 [38] initiates the design process by means of a "preliminary design stage" (Clause 5.2); ISO/IEC 12207:2008 [47] identifies the development of a system and software architecture (Clauses 5.3.2; 5.3.5); RTCA/DO-178C [25] again describes the first step of the software design process as the development of the software architecture (Clause 5.2.1a); whereas RTCA/DO-254 [26] uses the term "conceptual design" (Clause 5.2). The core concept in these descriptions is that a system or software structure is developed that enables the identification of system elements; the flow of information; and interfaces. Importantly, this high level design provides a framework for the allocation of requirements.

4.9.2.3 Low level or detailed design

Most of the references cited in the preceding section allude to a detail design activity or process. RTCA/DO-178C [25] refers to the development of "low level requirements". In the instance of airborne electronic equipment this low level or detailed design process refers to the design of printed circuit boards, software algorithms and hardware enclosures.

4.9.2.4 Realisation

Some references, e.g. ISO/IEC 15288-2008 [39], incorporate the realisation activity in a broader and encompassing implementation activity. ISO/IEC 12207:2008 [49] segregates this activity from the implementation activity, as software coding and (unit) testing. In order to separate responsibilities for tasks and activities, and due to different output validation mechanisms, it makes sense to classify the manufacturing of e.g. printed circuit boards, and hardware enclosures as realisation activities or process functions heading. It is feasible that not all items specified by the architectural or

detailed design process will be manufactured by the enterprise. Therefore, the specification of items to be procured, and the acceptance of procured items prior to integration into the system-of-interest by the project team, can also be considered to be a "realisation" activity.

4.9.2.5 Integration

All the references reviewed in this chapter indicated the need of an integration activity or stage. During the integration activity, at any layer of the system hierarchy, the elements or sub–systems that constitute the system at that layer, are integrated to produce the system, e.g. hardware or executable software code, at the hardware or executable software code layer. The integrated results from the lower layers are forwarded to the integration process function at the system-of-interest layer, where they are integrated to produce an artefact that is representative of the final version of the system, for verification testing.

During the integration testing process, tests are performed to ensure the correct operation of the system as well as to identify and remove errors and deficiencies. Dry runs of verification test cases and procedures should be executed during the integration process to ensure that the system will be accepted during verification tests, as well as to remove errors in the descriptions of verification cases and procedures.

Also note that in the case of platform specific airborne electronic equipment, the activities include air vehicle integration and flight testing. Therefore, the process must make sufficient provision to ensure the safety of the aircraft and its occupants during the flight tests while the airworthiness certification of the system-under-test is not completed.

4.9.2.6 Qualification, verification, and validation

The notions of qualification, verification, and validation are treated differently by most of the references consulted in this study. The concepts of qualification, verification, evaluation and validation, as discussed in e.g. ISO/IEC 12207, provide a basis against which these concepts can be examined.

ISO/IEC 12207 describes qualification testing, (Sub-clauses 5.3.9 and 5.3.11), which follows software and system integration as tests that are executed to demonstrate that a software product meets its specifications and is ready for use in its target environment. ISO/IEC 15288-2008 [39] describes a verification process (Sub-clause 6.4.6) that is also performed after completion of the integration process, which

demonstrates compliance to specified design requirements. As the verification tasks implied by ISO/IEC15288:2008(E) are not limited to tests, i.e. it can include analyses and reviews, the verification process described by this standard encompasses a wider scope than the qualification tests described by ISO/IEC 12207. However, it is evident that the intention of this technical process is the same as the qualification tests described by ISO/IEC 12207, i.e. to provide evidence of compliance to the system and software requirements.

- Validation

According to SAE ARP4754 [33], the term validation denotes "the determination that the requirements for a product are sufficiently correct and complete". RTCA/DO-178C [25] interprets validation to entail "the process of determining that the requirements are the correct requirements and that they are complete". It builds upon this notion by stating that "[the] system life cycle process may use software requirements and derived requirements in system validation".

The term validation is also used to express the process by which a system is evaluated in its target environment to prove that it fulfils its purpose. Validation also refers to the process when it is demonstrated that a (mathematical) model suitably represents the physical entity being modelled.

It will be shown in Chapter 4.9.4.2 that each output of every process function must be subjected to a process that ensures that the output is complete and correct, and that the process should provide corroborating evidence to the effect. The requirements formulations in an appropriate database must also be subjected to this process. Therefore, requirements validation should not be treated as a separate process function in the requirements process class.

- Verification

SAE ARP4754 [33] defines verification to be "the evaluation of an implementation of requirements to determine they have been met" and RTCA/DO-178C [25] construes it as "the evaluation of the results of a process to ensure correctness and consistency with respect to the inputs and standards provided to that process". An examination of these two definitions shows a semantic difference – ARP 4754 is more specific in terms of the relationship between the implementation of requirements and the verification activities, whereas DO-178C provides a more generalised interpretation.

It is also very important to note that a system can only be subjected to formal verification activities when a baseline is established that ensures that the integrity of

the lifecycle data describing the system at its current configuration is protected, as the verification results are associated with the system and all its lifecycle data.

- Segregation of the concepts of verification and validation

In this dissertation, the term validation firstly implies the process by which the correctness and completeness of lifecycle data and other process outputs are ensured. Requirements are to be validated prior to the commencement of the implementation of the requirements. For requirements that can be traced to requirements in higher layers of the system hierarchy, the traceability data serves as validation. For derived requirements, validation evidence is provided by documenting the rationale behind the requirement. The demonstration of compliance to a standard is another example of a validation activity. Other typical validation activities are design reviews and document inspections, where the objective of the review or inspection is to affirm the integrity of the design or document (as opposed to reviews that measure progress against a schedule, which is a project management activity).

Note that, according to the reasoning presented above, the general quality management practise of inspection and approval of lifecycle data can be considered as a validation activity.

In the process definition developed in this study, verification denotes the process of providing evidence that a requirement has been satisfied by an element (or arrangement of elements) of a synthesised system. Verification can be performed by means of tests, analyses and reviews. The verification method shall be selected on the basis of practicality and appropriateness. Demonstration that a synthesised system meets with a set of requirements is consequently a verification task.

This distinction between "validation" and "verification" is very important with respect to the process model that is derived in Section 4.7 and leads to significant simplification of the representation of the process.

The following concepts and terms apply to the verification and validation processes.

- Standard

RTCA/DO-254 [26] and RTCA/DO-178C [25] defines a standard to be "[a] rule or basis of comparison used to provide both guidance in and assessment of a given activity or the content of a specified data item". Standards include public domain standards, i.e. internationally and nationally recognised standards, and standards generated and maintained internally by the organisation, such as a requirements standard or design standard.

- Traceability

Traceability is formally defined in SAE ARP4754 [33] as "the characteristic by which requirements at one level of a design may be related to requirements at another level" and in RTCA/DO-178C [25] as "the evidence of an association between items, such as between process outputs, between an output and its originating process, or between a requirement and its implementation".

Cognisance must be taken of the fact that a confirmation of traceability between two items requires interpretation by a competent person to ensure that the association between the items is sound.

Traceability is imperative in safety critical systems since it must be shown on the one hand that no unspecified functionality is implemented in the final product as the system safety and verification processes are requirements based, and on the other hand it must be shown that all requirements were implemented. That is, a closed set of input requirements must be traced down to the lowest levels.

Also note that a lower-layer requirement can be considered to be validated if traceability to a higher-layer requirement is demonstrated if this higher layer requirement is validated.

- Coverage

The term "coverage analysis" is defined in RTCA/DO-178C [25] as "the process of determining the degree to which a proposed software verification process activity satisfies its objective" and RTCA/DO-254 [26] defines the term exactly the same but replaces "software" with "hardware".

As for the case of demonstrating traceability, confirmation of coverage requires interpretation by a competent person.

- Independence

RTCA/DO-178C [25] defines independence as the "separation of responsibilities which ensures the accomplishment of objective evaluation".

This recommended practise indicates the verification and validation activities that are to be executed with independence, depending on the software level. In practise it means that the same person who performed a task should not perform the verification and validation of the outputs of the task.

- Validation and verification methods and associated concepts

Three methods by which verification and validation can be achieved are identified, i.e. reviews, analyses and tests. The following definitions are summarised from RTCA/DO-254 [26]:

- A test confirms that the item correctly responds to a stimulus or series of stimuli. Testing performed for certification credit requires a configured item. Systems integration or software/hardware integration tests may also be used for test credit.
- An analysis is a detailed, repeatable analytical method for the evaluation of specific item characteristics to demonstrate that a specific requirement is met.
- A review is a qualitative method for the evaluation of the plans, requirements, design data, design concept or design implementation.

- Review and inspection

A review is a "qualitative evaluation to assess the plans, requirements, design data, design concept or design implementation to demonstrate to a high degree of confidence that the requirements have been or will be met" as defined in RTCA/DO-254 [26]. A distinction between reviews and analyses is clarified in RTCA/DO-178C [25] to be that "analyses provide repeatable evidence of correctness and reviews provide a qualitative assessment of correctness. A review may consist of an inspection of an output of a process guided by a checklist or similar aid."

The term review hence refers to all activities where the assessment of correctness is based on the opinions of subject matter experts. In the process developed as part of this dissertation reviews are notably used to validate outputs of activities, e.g. confirming the compliance of a design to requirements and standards, or the examination of a document to ascertain the correctness of the content and compliance with an applicable documentation standard.

An *inspection* entails "the examination and testing of supplies and services, including, when appropriate, raw materials, components, intermediate assemblies and services, to determine whether they conform to specified requirements", as defined in e.g. RTCA/DO-254 [26]. In the context of this study an inspection comprises a physical examination of an artefact or document. An inspection can form part of a review.

- Analysis

A definition of analysis is provided by RTCA/DO-254 [26] as "a process of mathematical or other logical reasoning that leads from stated premises to the conclusion concerning specific capabilities of equipment or hardware item and its adequacy for a particular application". With respect to software, RTCA/DO-178C [25] explains that an "analysis may examine in detail the functionality, performance, traceability and safety implications of a software component, and its relationship to other components within the airborne system or equipment".

- Assessment

The term assessment is addressed here as well, as assessments form important aspects of engineering processes, but should be distinguished from reviews and analyses. SAE ARP4761 [34] defines the term assessment to imply "an evaluation based upon engineering judgment". An example of the use of assessments is the system safety assessment process. Note that the assessment may rely on analyses to provide information on which evaluations are based. The major distinction between an analysis and an assessment is that an analysis uses a structured logical argument that may involve mathematical techniques, whereas an assessment relies on judgement based on experience and knowledge.

- Test

Tests can be conducted for a number of reasons, e.g. to improve the understanding of requirements, to investigate implementation alternatives and to support verification processes. The following description pertains to *verification* tests.

A test comprises "a quantitative procedure to prove performance using stated objective criteria with pass/fail results", as expressed in RTCA/DO-254 [26].

In terms of software testing, the objectives of the software testing process is stated in RTCA/DO-178C [25] to firstly demonstrate that the software satisfies its requirements and secondly "to demonstrate with a high degree of confidence that errors which could lead to unacceptable failure conditions, as determined by the system safety process, have been removed". Elements of the test process for software are defined in this standard as follows:

A test case is "a set of test inputs, execution conditions and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement".

Testing denotes "the process of exercising a system or system component to verify that it satisfies specified requirements and to detect errors".

A test procedure is a compilation of "detailed instructions for the setup and execution of a given set of test cases, and instructions for the evaluation of results of executing the test cases".

Note that the above definitions can be extended to system testing as well.

4.9.2.7 Support system development

The segregation of a system into end products and enabling products was described in section 4.4. The development of the support systems for an airborne electronic system, e.g. D-level test equipment; maintenance schedules and procedures; operating instructions; and training material needs to be addressed in the process design as well.

4.9.2.8 Prototyping and simulation

Prototypes and simulations are important elements of systems engineering processes. A prototype is defined by ANSI-EIA-632 [12] as:

> "[A] model, (physical, electronic, digital, analytical, etc.) of a product built for the purpose of a) assessing the feasibility of a new or unfamiliar technology; b) assessing or mitigating technical risk; c) validating requirements; d) demonstrating critical features; e) verifying a product; f) validating a product; g) determining enabling product readiness; h) characterising performance or product features; or i) discovering physical principles.

In terms of this definition, computer based simulations can also be viewed as a class of prototype, as simulations are typically digital analytic models developed for one or more of the purposes described in the definition. Prototypes also include software modules produced with objectives that can be associated with this definition.

The use of prototypes or simulations with regard to the development of airborne electronic systems are described below, under headings derived from the definition given above.

a) Assess the feasibility of new technology

It was pointed out in Section 1.1 that the technologies employed in the realisation of electronic systems for airborne use are subjected to rapidly changing technologies.

It may be required to develop prototypes to assess the use of new technologies, or technologies with which the development team are not familiar. These prototypes will typically implement partial functionality, and airborne testing of the prototype will be performed under stringently controlled conditions. The prototype system will not be subjected to all the airworthiness process described in Section 4.7.

b) Assess or mitigate technical risk

The use of prototypes to assess technical risks, in the context of the systems described in this study, is similar to the use of prototypes to assess the feasibility of unfamiliar technologies, as described in the previous paragraph.

c) Validation of requirements

A significant use of prototypes, including simulations, is in the validation of requirement statements, when no other means of validation is feasible. Prototypes or simulations that implement such requirements are developed to a sufficient level of fidelity as to enable the validation process.

Prototypes are also developed to support the process of identifying and understanding requirements in instances where requirements are not fully developed at the outset of the project. By implication, this use of prototypes can also be viewed as requirements validation.

d) Demonstration of critical features

For platform-specific systems, i.e. systems where there are no marketing processes associated with the development of the system, it is not envisaged that it will be a requirement to build a prototype to demonstrate critical features of the product.

e) Product verification

The concept of development of a first article from the outset was explained in the retrospective assessments. It was also identified that verification should only be performed on a fully representative product, i.e. typically a first article. Therefore, for a prototype that was not subjected to all the controls as in the case of a first article, the verification process cannot be viewed to be valid and hence this instance of the use of a prototype does not apply.

f) Product validation

The use of prototypes for product validation does not apply for the same reason as stated above.

g) Determining enabling product readiness

In isolated cases, a prototype might be required to determine the readiness of enabling products, but this use of prototypes is not assessed further in this section.

h) Characterising product features or performance

If the product may have features or performance characteristics that did not result as consequences of the specification or design processes, it may be required to produce a prototype to evaluate these features. For the type of systems described in this study, this is however a very unlikely scenario, and this instance of the use of prototypes is also not considered further.

i) Discovering physical principles

Equipment developed for use on board aircraft intended for series production generally rely on well-established physical laws and principles. The likelihood of prototypes being developed for the investigation of physical phenomena in this scenario is highly improbable, and this topic is hence not assessed.

For the development of airborne electronic equipment, it can therefore be stated that prototypes will primarily be developed to improve the understanding of aspects concerning the system to be developed, in terms of technologies, risk, and system properties. An important aspect of such prototypes, in contrast to qualified products, is that the rigour of control over the development of a prototype is not necessarily at the same level as that which is required for a qualified airworthy product. Importantly, prototypes developed with any of these three objectives in mind must have the ability to rapidly incorporate changes in terms of requirements, as well as in design and implementation aspects, without compromising the integrity of relevant data.

It is important to consider the appropriate use of prototypes, as well as the methods associated with the development of prototypes in the establishment of a documented development process.

The fundamental difference is that a first article represents a qualified product, while prototypes are not qualified, i.e. subjected to comprehensive verification and validation processes.

4.9.3 Development process framework

A generic lifecycle model for a process for the development of airborne electronic equipment commensurate with contemporary airworthiness principles is developed in this section. This is done by revisiting the analysis of RTCA/DO-178C [25] as

developed in the previous section and deriving a condensed abstracted formulation of the processes described in the standard – that is, the process philosophy. This abstraction is then generalised and applied to development of electronic hardware as well as to the integrated electronic sub-system.

4.9.3.1 First article concept

As described in Section 4.7.4, the artefact representing the verified system-of-interest that the development process generates is fully representative of the "production" units. This differs from more conventional views where a series of prototypes were developed towards the final product. This methodology is feasible due to the specific nature of modern electronic equipment containing embedded software. The use of modern design tools makes it possible to design enclosures and electronic boards, and to validate these designs prior to manufacture, without resorting to physical prototypes. The software development process prescribed by e.g. RTCA/DO-178C also motivates a top-down process, without resorting to "prototyped" code. It was shown in Section 4.7.7.2 that the artefact produced by the formal development process is a "first article". Therefore, it is important that the lifecycle model that is derived in this section accommodates this "first article" concept.

4.9.3.2 Utilisation of Process Functions

It was shown in Section 4.4.9 that the development process elements, also referred to as "activities", can formally be represented as sets of interconnected process functions. Each process function acts on a set of inputs to produce a set of outputs, and each process function is supported by a set of enablers, and subjected to a set of controls. Outputs of one process function become inputs to other process functions.

This concept of "process functions" provides a convenient basis for the description of the activities that constitute a development process.

4.9.3.3 Generalisation of the lifecycle processes

A significant generalisation can be made when it is considered that the activities constituting the development lifecycle processes, as described in the foregoing sections, can be classified into three main activity focus areas, i.e. requirements; synthesis; and verification processes, as follows:

- The objectives of the requirements processes are to determine the functional and performance requirements of the system-of-interest, and to identify

implementation constraints. Other required attributes of the system-of-interest are also defined. All these are to be described clearly and unambiguously as a set of functional and non-functional requirements.

- The synthesis processes develop architectural and detailed designs and generate and/or procure hardware elements and software code, i.e. these processes implement the requirements. The processes also include all hardware and software integration actions to produce the system characterised by the requirements.
- The purpose of the verification processes is to obtain objective evidence to prove that the system, as produced by the synthesis process, meets with the requirements produced by the requirements process.

Development is concluded when verification evidence with respect to contractual and certification requirements is formally accepted and all discrepancies have been resolved. In terms of this simplification, the process representation can be reduced to the schema shown in Figure 56.

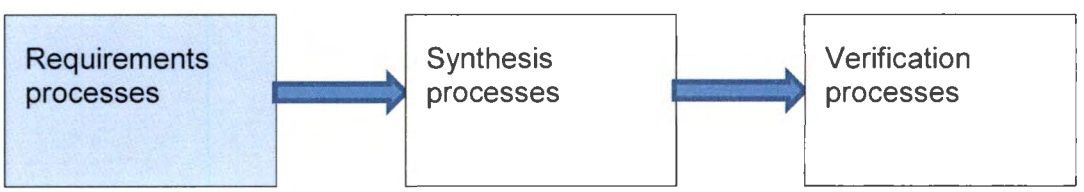

Figure 56: Process groups

In Chapter 5.2.1 it will also be shown that this process structure provides a convenient basis for the establishment of baselines that allow for incremental and iterative development of functionality.

The grouping of these processes according to this breakdown, as they are presented in some of the contemporary standards reviewed in the preceding sections, is shown in Table 13.

Table 13: Generalisation of life cycle processes

Standard	Requirements processes	Synthesis processes	Verification processes
EIA-632	Requirements management	Solution definition	Verification
		Implementation	
ISO/IEC 15288:2008	Stakeholders requirements definition	Architectural design	Verification
	Requirements analysis	Implementation	
		Integration	
ISO/IEC 12207	System requirements analysis	System architectural design	System qualification testing
		System integration	
	Software requirements analysis	Software architectural design	Software qualification testing
		Software detailed design	
		Software coding and testing	
		Software integration	

4.9.3.4 Associations with the system safety process

Considering the system safety process as described in Section 4.7.3, the associations as shown in Figure 57 can be made.

- Association of FHA with the requirements process

The FHA as described in SAE ARP4761 [34] reviews the functional requirements of the system, including derived functional requirements, to determine the severity of the consequences of failures or the specified functions. These severities define design objectives that aim at mitigating the risks associated with the identified functional failures. Therefore, it is important that any changes to functional requirements, as well as new derived requirements, should be subjected to the FHA process. Therefore, the FHA should be performed in association with the requirements process.

- Association of Preliminary System Safety Assessment with the high level design process

The PSSA as prescribed by the system safety process, described in SAE ARP4761 [34], evaluates the system architectural design to determine whether the design contains sufficient redundancy, as well as to determine whether the development assurance levels are commensurate with the safety objectives as determined by the FHA.

- Association of System Safety Assessment with the verification process

The objective of the SSA, as it is described in SAE ARP4761 [34], is to verify that the safety requirements were met.

These associations are shown in Figure 57.

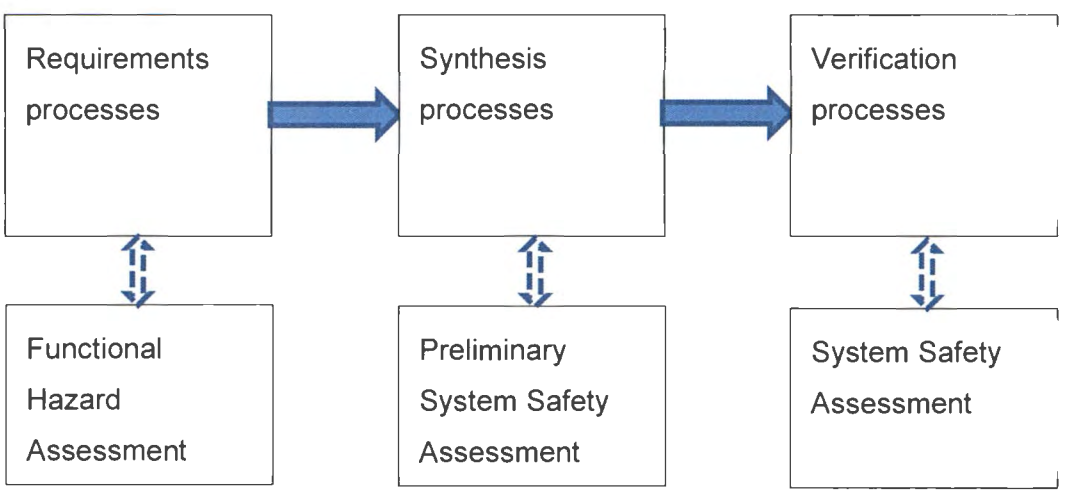

Figure 57: System safety activities associated with process groups

4.9.3.5 Process representation

The development process model shown in Figure 56, generic to any layer of the system hierarchy, can be represented in a high level process function presentation as shown in Figure 58. Each process function receives inputs and produces outputs, and each process function (or selected group of process functions) is subjected to controls, and the process function is executed using a set of mechanisms or enablers.

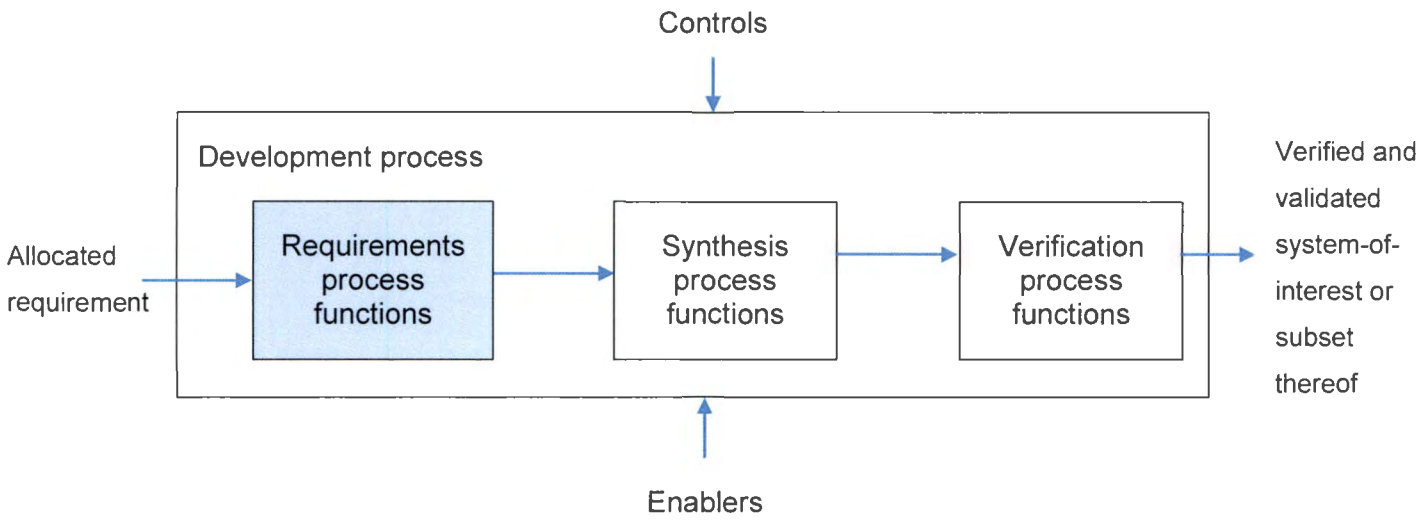

Figure 58: The development process model in process function building block representation

4.9.3.6 Generalisation of the RTCA/DO-178C lifecycle model

The lifecycle model portrayed by RTCA/DO-178C [25] is described in Section 4.7.4. This standard is the de facto reference for airworthiness approval of airborne software. It is thus logical that any process design should be commensurate with this standard. In discussing the lifecycle processes represented by RTCA/DO-178C, a simplified lifecycle model was derived, as shown in Figure 45. This abstracted software development lifecycle concept can be extended to the design of the electronic system (that contains embedded software) as well. This generalisation is shown in Figure 59. Also shown in Figure 59 is the breakdown into the process groups, as described in section 4.9.3.3. It is also important to note that in this diagram, the outputs of the process are validated at the output stage of each process function.

At the highest level of the breakdown of the process structure, the generalised process view presented in Figure 59 is redrawn, as shown in Figure 60, in terms of the structure depicted in Figure 58.

The process description presented in Figure 60 allows for the organisation of all the tasks and activities, as summarised in Section 4.9.2, into the following categories:

- Requirements processes;
- Synthesis processes;
- Verification processes;
- Control processes;
- Process enablers.

Process enablers are described in the next section.

The control processes are described in Section 4.9.4, where the solutions to the fourth research challenges are addressed.

.

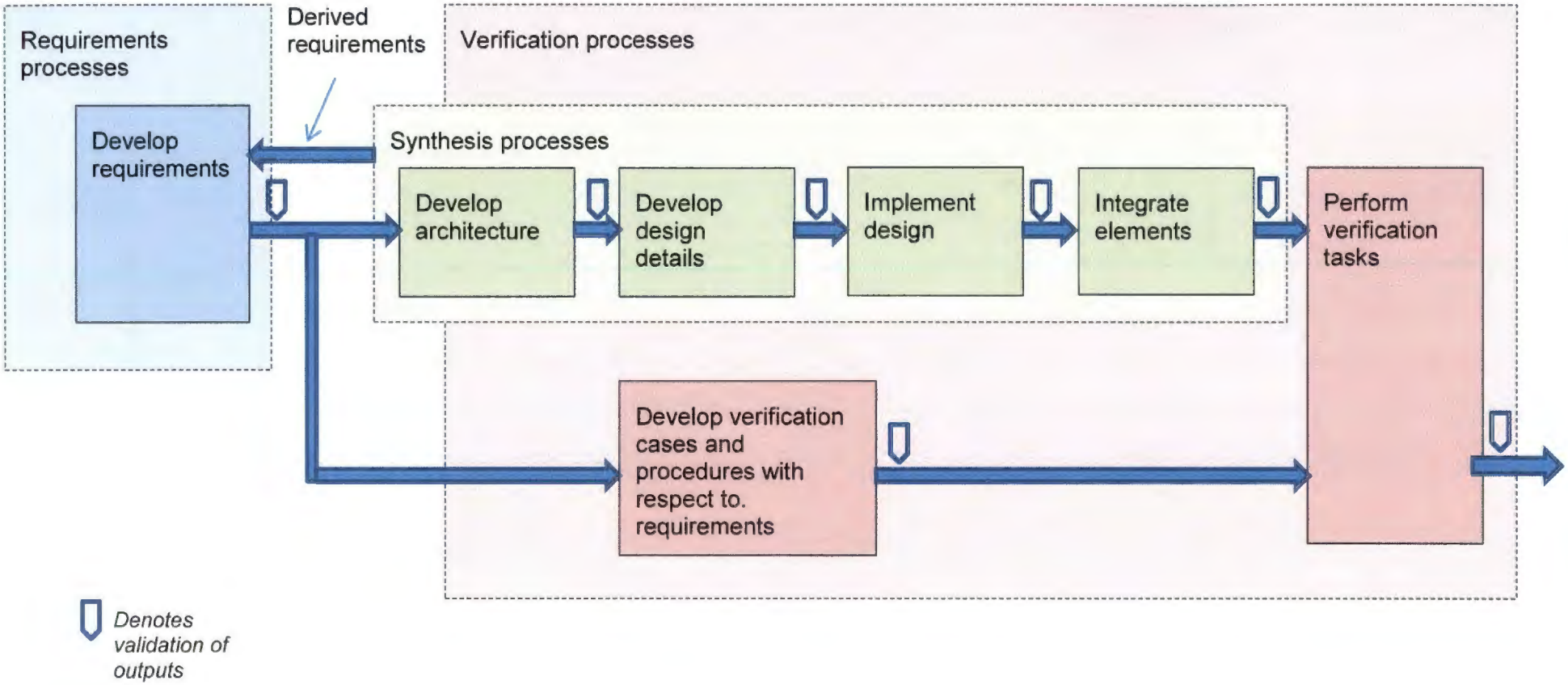

Figure 59: Generalised view of the RTCA/DO-178C process

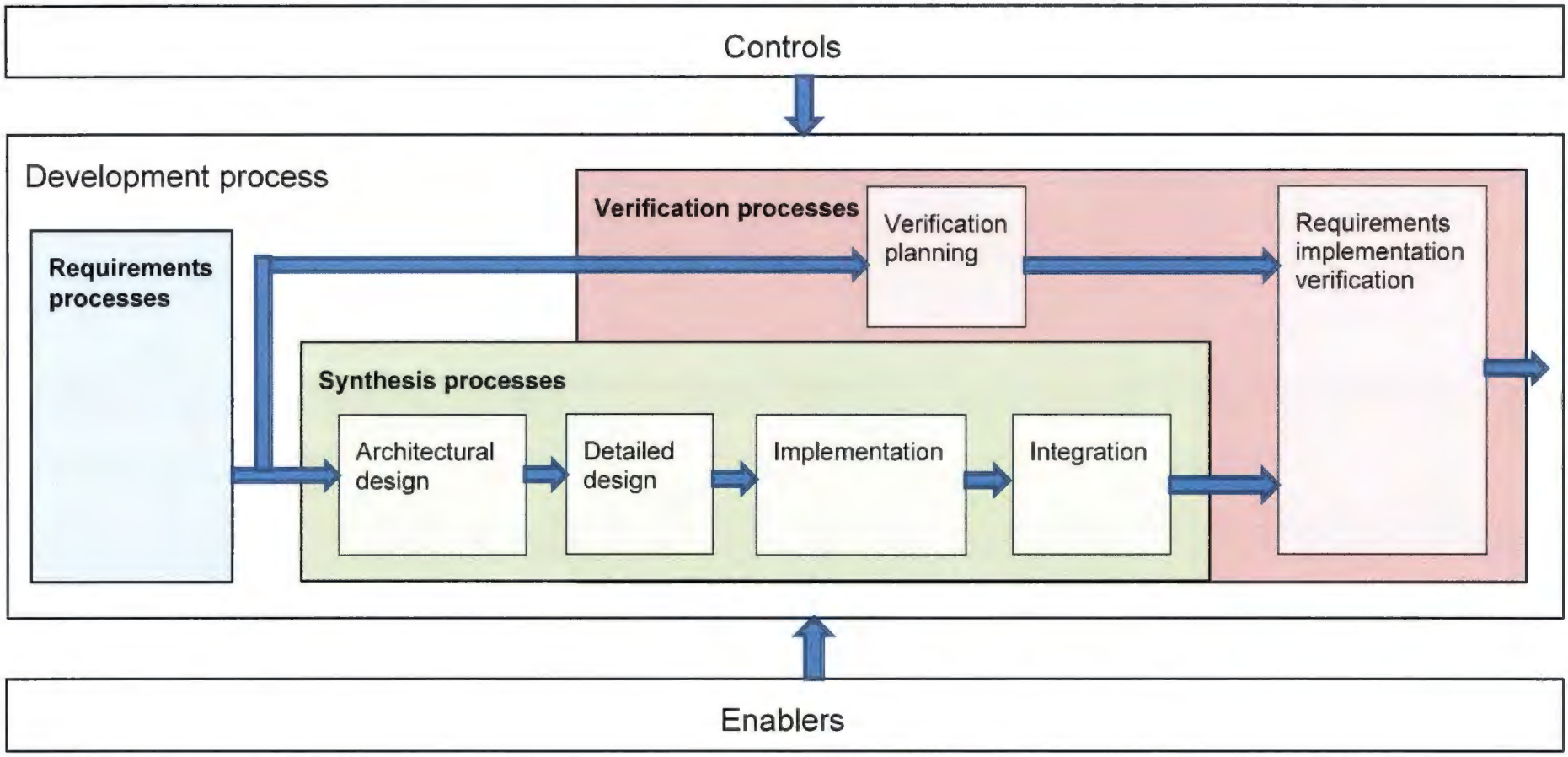

Figure 60: High level life cycle model

4.9.3.7 Process enabling mechanisms

In the context of this lifecycle model, enablers are the mechanisms that facilitate the execution of a particular process function.

- Simulation, modelling and prototyping as enablers

Concepts such as rapid prototyping and simulation can also be used in support of e.g. refinement of requirements or testing design trade-offs. Considering the definition of enabling systems from ISO/IEC 15288-2008 [39], which states that an enabling system "supports a system-of-interest during its lifecycle stages, but does not necessarily contribute to its function during operation", prototypes and simulation can be viewed as entities of enabling systems, as long as they do not form part of the formal definition of the product.

- Configuration Management as an enabling mechanism

Although CM is identified as a control mechanism, aspects of the CM system can also be considered as supporting mechanisms, i.e. storage and management of document templates, and the assurance of availability of accurate project data from a central data repository.

- Methods and tools as enablers

It also follows from the reasoning provided by Estefan [43] that methods and tools resort under the classification of enablers or mechanisms. This is a straight-forward conclusion and does not merit further explanation.

Methods are described in the QMS (see Chapter 4.5.1), hence the QMS can be viewed as a primary enabler. Additional plans that are prepared particularly for the project, which also may contain descriptions of methods, are also classed as enablers.

Tools include PLM systems, computer aided design tools, integrated development environments for developing software code, and integration and test systems.

- Planning and planning documents as enablers

All the systems engineering and related standards quoted in this dissertation require the establishment of development plans. As an example, ISO/IEC 15288-2008 [39] states that the purpose of the project planning process is "to produce and communicate effective and workable project plans". IEEE STD 1220-2005 [38] requires an engineering plan that "shall be prepared an updated throughout the system lifecycle to guide and control the technical efforts of the project".

These technical plans provide guidance in terms of methods and procedures to be followed, and importantly, define or identify the criteria against which the outcomes of tasks can be

evaluated, hence enabling the validation process. Note that these plans include certification related plans, therefore implicitly the certification liaison process can be considered as an enabling process.

The relationship between the process description developed in this study and the technical plans developed for each project is described in Chapter 4.5.3.

4.9.3.8 Iterative and recursive application of the process model

In the process model derived above, the typical use of an iterative and recursive approach will be to determine and validate the requirements for the system-of-interest and to then select a limited series of subsets of the requirements that will be synthesised and verified, until the complete set of requirements are implemented and verified.

When the synthesis processes for block 1.0 are complete, a block 1.0 verification baseline is established, which then prohibits any unauthorised changes to any lifecycle data. Block 1.0 is then subjected to the verification process, and upon successful completion the block 1.0 product baseline is delivered. This product baseline is demonstrated to have fully complied with the requirements, and lifecycle data, e.g. source code that is associated with this block may not be altered unless a new development increment with respect to its functionality is authorised. This philosophy is applied incrementally until all the predetermined blocks are verified. The concept is shown diagrammatically in Figure 61.

Importantly, also note that an iterative process is also followed within the synthesis process of any block, as implementations are tested during integration testing to resolve design or manufacturing problems and that the problems are rectified before the establishment of the verification baseline. This mechanism resolves significant problems regarding feedback of information, which may be required within the synthesis and verification cycle.

The process model is recursive, as it is applicable to all the layers of the system hierarchy.

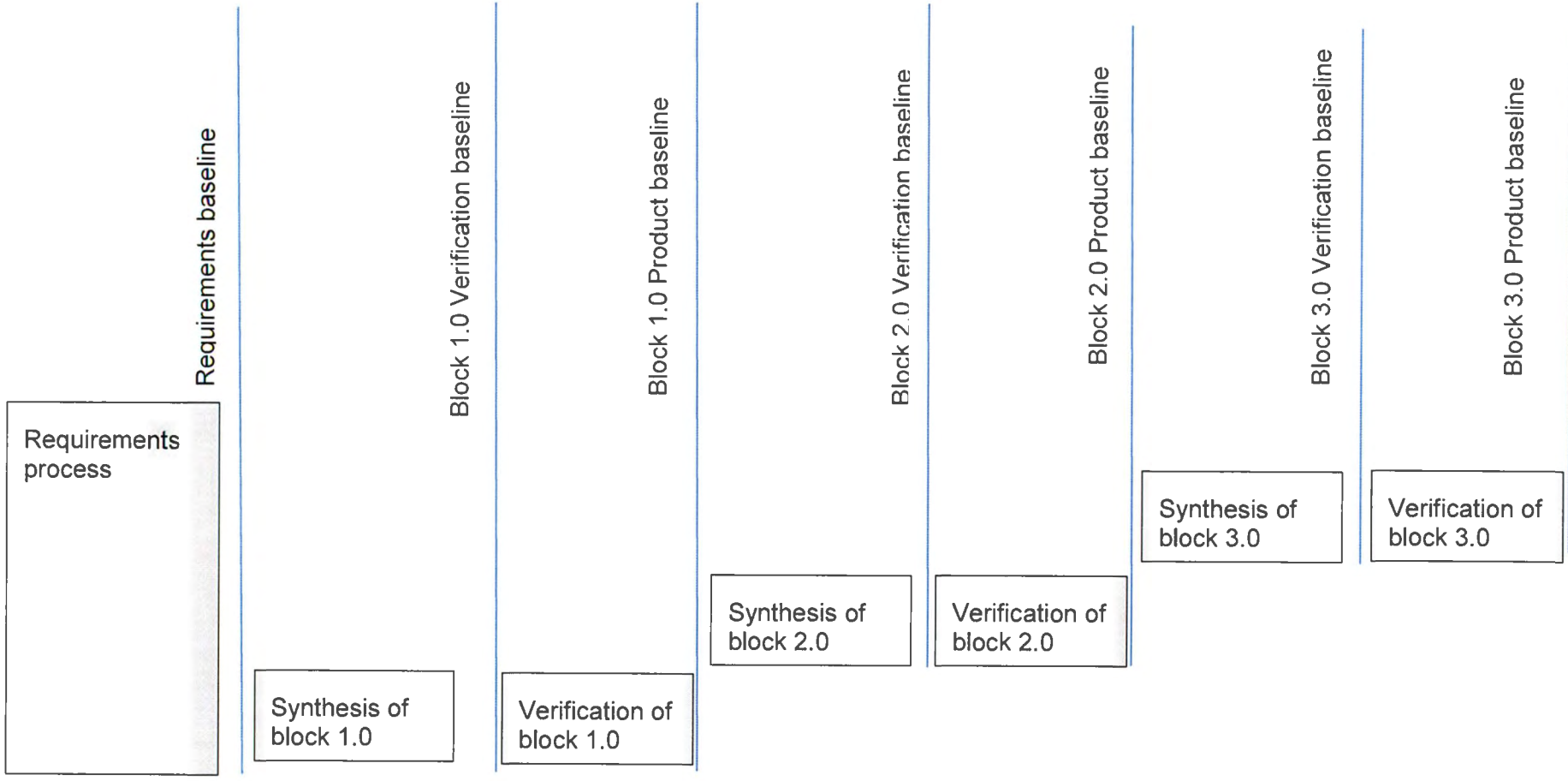

Figure 61: Iterative and recursive development

- Concept stage iterative loops

It is important to recognise that the agreement can only be finalised when the system architecture (including interface protocols and redundancies) is defined and when the design assurance and software levels of the constituent CIs were determined, and the extent of verification testing is known, as these are the most significant cost drivers. This implies that the product configuration data associated with the Requirements Analysis, Functional Hazard Assessment, Architectural Design, Preliminary System Safety Assessment and Verification Planning activities can only be baselined after all the iterations within this loop have been completed. This approach allows for a significant reduction in the project risks in terms of costs estimation, as well as the containment of "scope creep".

The requirements should be baselined, i.e. protected against unauthorised changes, after finalisation of the agreement.

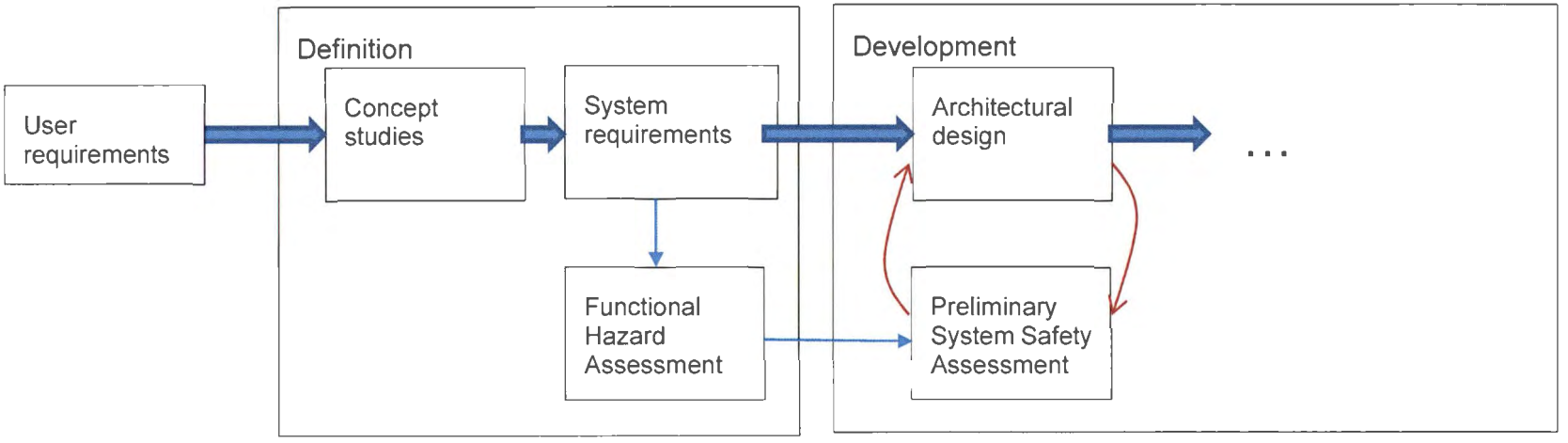

Figure 62: Concept stage iterations

4.9.4 Development process controls

The objectives of the development process controls are firstly to ensure that authorised tasks are performed with respect to lifecycle data items of interest and to manage the integrity of the data and, secondly, to confirm that they have been executed correctly upon completion of these tasks.

4.9.4.1 Configuration management as a process control mechanism

An established configuration management process, such as described in section 4.6, will meet with the first control objective stated above. In particular, a change control process associated with a specific baseline structure and change authorisation methodology will ensure that no unauthorised changes will be performed on controlled lifecycle data and that appropriate changes will only be implemented on correctly identified data items.

4.9.4.2 Assurance of correctness of process function outputs

The second objective is achieved through activities that ensure that the outputs of activities meet with requirements and standards applicable to those activities. In terms of the discussion presented in section 4.9.2, these are the validation processes associated with specific tasks and outcomes. These validation tasks include all the activities aimed at confirming the correctness of outputs of process functions, and the generation of evidence of the validation results.

Note that by defining the process in terms of a logical combination of process functions, and ensuring the comprehensive validation of the outputs at the level of each process function, objectives from quality assurance and systems engineering viewpoints can be unified. By ensuring the participation of appropriate team members in these validation tasks, the objectives of independence can also be achieved.

If the validation tasks are performed with sufficient independence, the objectives of quality, process and design assurance as recommended by systems engineering and airworthiness references, are implicitly achieved.

The efforts required to comply with process requirements from the various stakeholders can be consolidated by merging of the validation tasks for each process function. Records of the validation process are provided to aircraft layer processes, e.g. quality assurance, configuration management and airworthiness management.

4.10 Summary of the literature study

The subject matter literature that formed part of this study was divided into specific areas of interest to organise the study. In each of these areas of interest, important individual concepts were identified and analysed. The concepts were assessed in terms of their usefulness, or lack thereof, for implementation into the development process design. A summary of the literature study is presented below. A graphical depiction of the literature study is presented in Figure 63.

Literature group	Motivation for inclusion in study	Summary and salient aspects	Concepts applied / discarded
Classical references • Blanchard & Fabrycky • RSA-MIL-STD-3 • MIL-STD-498	• Widely referenced in defence industry • Legacy with respect to. applied processes	• Highly prescriptive • Project management and technical objectives not separated • Use of specific prototypes • Characterised by specific baselines, reviews and milestones • Hierarchical product structure • "Waterfall" approach	Apply: • "Classical" standards referenced in development of checklists • Use standards in validation of process design Discard: • No "classical" concepts included in process design
Contemporary systems and software engineering standards • IEE 1220 • ISO/IEC 15288 • ISO/IEC 12207 • ANSI/EIA 632	• Widely used for guidance in planning of present-day development projects	• Describe generic lifecycle concepts • Separation of technical and project objectives • System-of-systems approach • Focus on "how" rather than "what" • End product and enabling product concepts	Apply: • Separation of technical and project objectives • System-of-systems approach • Focus on "what" rather than "how" • End product and enabling product concepts • Aspects from descriptions of task details • Use in validation of process design
Quality management standards • ISO 9000/1 • SAE AS9100	• Organisations developing airborne systems must comply with these	• Require a process approach to ensure consistency of products and services • Describe aspects of engineering tasks (product realisation) • Common purpose exists with SE principles	Apply: • Process approach • Aspects from descriptions of product realisation task details • Quality management as a process control mechanism
Configuration management standards • ISO 10007 • EIA-649	• Widely established methodology used in military and aerospace industries	• Describe principles and methods to control the development of lifecycle data and to manage the integrity of this data	Apply: • Configuration management as a process control mechanism • Baseline management principles • Control categories
Airworthiness recommended practises • SAE ARP4754 • SAE ARP4761 • RTCA/DO-178B/C • RTCA/DO-254	• Establishes airworthiness principles to be incorporated in process design	• Invoke system safety processes • Specify a requirements based process • Describe appropriate lifecycle models • Describe the concept of development assurance	Apply: • Process design is based on a lifecycle model derived from these recommended practises
Academic literature • INCOSE handbook • SEBoK • Journal articles	• Provide further perspectives on engineering process lifecycles and principles	• Different lifecycle model types described • Support notion of difficulty to establish requirements up front • Support notion of difficulty to estimate scope of work accurately	Apply: • Pre-specified, multi-step model • Iterative and incremental development Discard: • Agile methods

Figure 63: Literature study summary

4.11 Conclusion

The literature study presented in this chapter covered literature focus areas relevant to research challenges defined in Chapter 3.

Further validation of the research problem was obtained from a review of the literature in specialty fields.

In summary, literature focus areas of this research are shown in the columns of Table 14, with the research challenges, as defined in Figure 13, shown in the rows. This table is used to link a research focus area to each research objective, where the information from a literature focus area is applied to address specific research problems as indicated in the table. Research problems are then translated to research solutions by using the information from the focus area. Research solutions are shown in the bottom row of Table 14.

Thus, even though each separate research objective can be addressed by an identified solution, an integrated solution was not fully dealt with in literature.

A lifecycle model applicable to the development of airborne electronic equipment containing embedded software which can be implemented directly is not explicitly defined.

Table 14: Literature study focus areas applicable to the research challenges

Literature focus areas / Research Challenges	The scope of the engineering process for the development of airborne electronic equipment is not pertinently established.	The activities and tasks required for the development of platform specific airborne electronic systems are not identified explicitly.	A lifecycle model suitable for the development of airborne electronic equipment, in particular resolving the problem of information feedback, is not determined.	The development process control mechanisms are ineffective.
Classical systems engineering approaches and standards	↑	↑	↑ ↓	↑
Contemporary systems engineering standards	↑	↑ ↓	↑ ↓	↑
Quality management requirements and considerations	↑	↓		↓
Configuration management considerations				↑ ↓
Airworthiness requirements and recommended practises	↑ ↓	↓		
Lifecycle models described in academic literature			↑ ↓	
Literature focus areas / Research Solutions	A process context is delineated for airborne electronic equipment.	Generic activities and tasks required for the development of airborne electronic equipment are identified.	An appropriate lifecycle model for the development of airborne electronic equipment is identified.	Suitable development control mechanisms are identified.

Legend: ↑ - Literature focus area validates the research challenge

↓ - Literature focus area contributes to the research solution

Chapter 5 Development and validation of a process design

In Chapter 2.6 it was indicated that, in terms of the design science research method, the process design which is developed as a result of this study is an improved solution to a known problem. It must be noted that the process design presented in this chapter is a solution, but not a unique solution.

The first step in the design of the process is to define the context and boundaries of the process in order to demarcate the scope of the process design.

5.1 Detailed contextualisation of the technical processes

The technical processes for the development of a system were differentiated from the other elements of the project processes, as shown Chapter 4.4.9. This segregation of the technical processes, as determined by the DSR rigour cycle, provides a basis for the outlining of the scope of a process for development of platform specific airborne electronic equipment.

The context of the technical processes within the greater operating environment is shown in Figure 64. In this representation, the technical processes only interact with project processes and with the airworthiness certification authority. The interfaces between these entities are shown in Table 15.

The technical processes encompass the activities of technical planning and management, requirements management, system design, prototyping and modelling, procurement and integration of first article components, software coding, system and software integration, and verification. These activities include speciality areas such as airworthiness, safety, reliability, security, maintainability, supportability, human factors, electromagnetic compatibility (EMC), flight testing and application of other subject matter expertise. The controls applicable to these processes, e.g. quality assurance and configuration management, are also included.

In this representation, project management processes are concerned with co-ordination of activities between the client and engineering processes in terms of the agreement, schedules, and contracting of resources and facilities from the enterprise support functions.

The flow of information between the technical processes and the project processes is as follows:

The Statement of Work (SoW) is developed by the technical processes based on information provided via the project processes. A schedule is developed by the project processes based on the SoW and the work is allocated to the technical team by means of work authorisations. The technical processes provide deliverables to the project processes against the schedule and in accordance with the work authorisations.

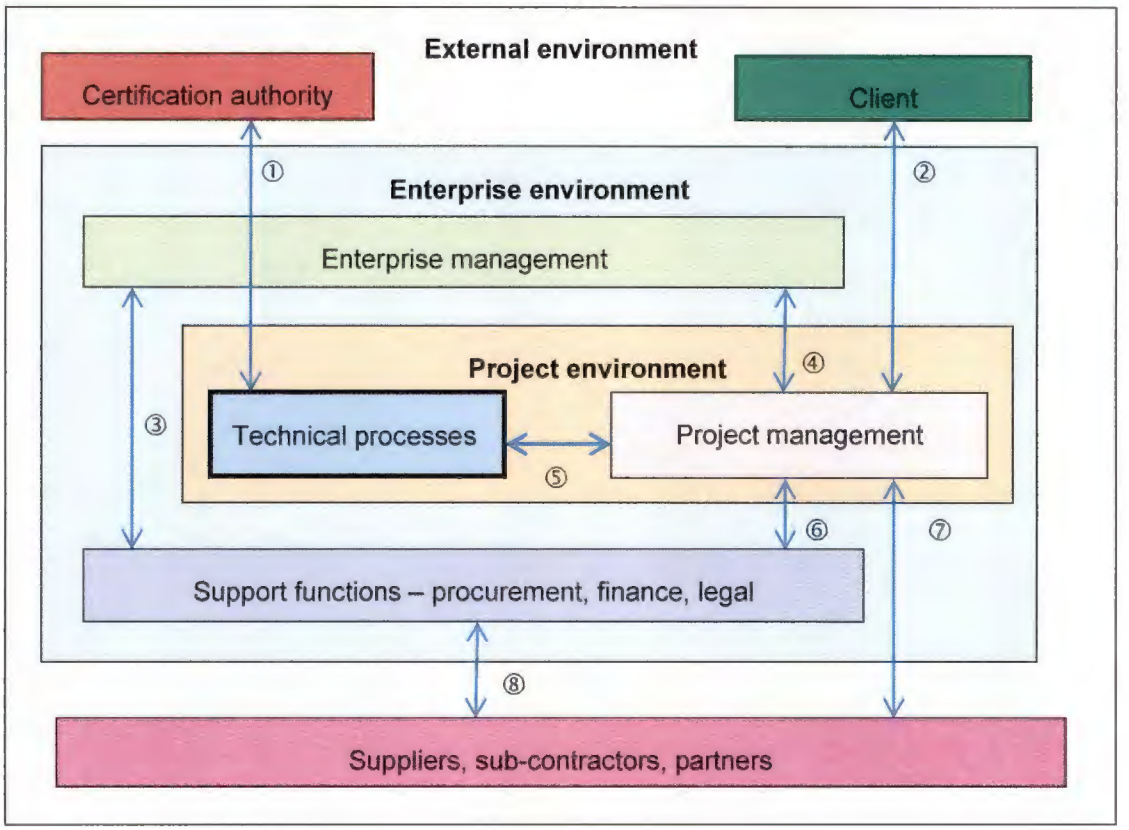

Figure 64: Process context

The literature study has shown that the technical processes can be further be segregated into development and lifecycle support processes; the focus of this study is on the development processes. The outcome of the activities of the development processes is a verified and certified product and its associated enabling products, as well as the identified product configuration information.

Table 15: Organisational interfaces

No	Interface	Interface subjects
1	Technical processes – Certification authority	• Certification basis • Certification requirements • Certification data
2	Client – Project management process	• Schedules • Project reviews • Progress reports • Deliverables • Invoices • Payments
3	Enterprise management – Support functions	• Operational communications
4	Enterprise management – Project management process	• Schedules • Progress reports • Corporate level interventions
5	Technical processes – Project management processes	• Statement of Work (SoW) • Schedule • Work authorisations • Deliverables
6	Project management processes – Support functions	• Requirements • Schedules • Progress reports • Deliverables • Payments
7	Project management processes – Suppliers, sub-contractors, and partners	• Requirements • Schedules • Progress reports • Deliverables • Payments
8	Support functions – Suppliers, sub-contractors, and partners	• Requirements • Schedules • Progress reports • Deliverables • Payments

5.2 Development of a process framework

The lifecycle model for the process is outlined in the following section. This is a "fleshing-out" of the lifecycle model derived and validated in Chapter 4.9.3.

5.2.1 Baselines types

The concept of a baseline was explained in Chapter 4.6.2. The baselines to be used in the process design are described in this section, considering the three main process groups constituting the lifecycle model, shown in Figure 56.

5.2.1.1 Requirements baseline

Once the requirements for a system are validated, these requirements form the references for the synthesis and verification processes. It follows that the validated requirements statements should not be altered indiscriminately, as it will affect traceability between the requirements and the implementation. Therefore, the requirements should be under formal change control from the time validation is completed, which implies the baselining of the requirements by means of the configuration management process. For the purpose of the process design described in this study, this baseline is termed the requirements baseline.

5.2.1.2 Verification baseline

The synthesis process group produces the integrated system, containing hardware and software as well as enabling products. Upon completion of the synthesis process, the system is subjected to the verification process to demonstrate compliance with the requirements. It is important to understand that the system or the lifecycle data describing its attributes cannot be modified while the verification process is in progress. If the design of the system is altered in order to correct a deficiency, some system attributes may change, which could imply that the behaviour of the system may have been affected and that previous test results on the system may be rendered invalid. Therefore, it is prudent to ensure that the system contains no defects at the stage when it is subjected to verification tests.

The practical approach is to subject the system to dry runs of the verification tests before the formal test is performed as described in Chapter 4.9.2.5. These dry run tests may detect latent defects that were not detected during integration testing and may also identify errors in test procedures, which can be corrected prior to formal

verification testing. At the point where the validation of all the outputs of the synthesis process, as well as the validation of the test cases and procedures have been completed, the verification baseline is established.

5.2.1.3 Product baseline

Following the establishment of the verification baseline, the system is now subjected to the formal verification tests to show that the system meets with the requirements. The product baseline is established upon completion of the verification tests. Also note that the product baseline

The product baseline represents the system when it is declared fit for operational use, and this baseline is referenced in the aircraft's Certificate of Design (CoD).

5.2.2 Workflow considerations

ISO/IEC 12207:1995 [47] identifies a workflow as a logical concatenation of activities. The sequence in which tasks are executed affects the implementation of the baseline control process. These considerations are discussed below.

5.2.2.1 Design information feedback loops

The handling of information feedback in the development cycle was described in Chapter 4.3.3.3 and in Chapter 4.9.3.8.

Feedback loops occur when information from a "downstream" process function requires changes to "upstream" process functions. A notable problem with design information feedback loops is the fact that baselined configuration data cannot be changed easily. The different feedback loops that can occur during the development lifecycle are considered in this section.

The three process groups constituting the lifecycle model which was derived in Chapter 4.9.3.6 are shown in Figure 65. Also shown in Figure 65 are six feasible information feedback paths that can occur during a development cycle.

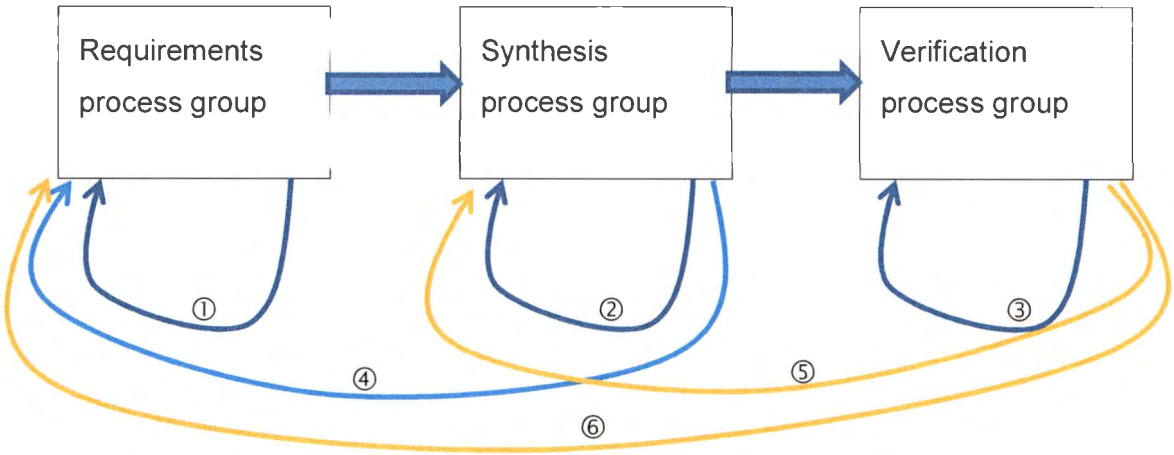

Figure 65: Design information feedback loops

The first feedback loop exists within the requirements management process group and occurs while establishing requirements. During the activity of formulating requirements, they are reviewed and updated up to a stage where all requirements relevant to a specific project are validated. During this period, it should be possible to change requirements statements (e.g. in a database) without undue constraints. Therefore, the mechanism used to record requirements should allow for changes during the requirements formulation cycle, but thereafter, i.e. after establishment of the requirements baseline, it must be possible to prevent unauthorised changes to requirements statements to protect the integrity of the validated requirements.

The second loop shown in Figure 65 is the loop within the synthesis process group which is inherent to the design and integration process. It was pointed out in the retrospective assessment that feedback loops exist between the Detailed Design, Design Realisation and Integration process functions. Problems and errors are invariably detected during the integration activities. These problems typically need corrective action in terms of a realised system element and the lifecycle data associated with the element. This implies that the control of the lifecycle data should allow sufficient flexibility to allow changes to be made promptly, up to the establishment of the verification baseline.

The third loop, which exists in the verification process group, involves the development of verification cases and procedures. Integration testing must include dry runs of verification tests, in order to remove errors from the subject under test, as well as from

test documentation. Adherence to this notion will prevent undue rework during the formal verification stage. Draft issues of test documentation should be used to ensure that all errors are removed from the test procedures prior to formal verification testing. This action also serves as validation of the test procedures. Note that this loop also ceases to exist after establishment of the verification baseline.

The loops described up to this point can be considered to be "normal" for an engineering process and pose no challenges in terms of a process design.

The fourth loop shown in Figure 65 occurs when either new requirements were identified, or errors were found in baselined requirements, as a consequence of synthesis activities. In this instance, corrections to requirements require the update of the requirements baseline, which can only be performed by applying the formal configuration change control process.

The fifth and sixth loops shown in Figure 65 can be considered together. These loops occur when a problem was detected during the verification process, which necessitate updates to the verification baseline or the requirements baseline, or both. A primary objective of the formalisation of the development process is to prevent the occurrence of these loops, due to the consequences in terms of the updating of the requirements and verification baselines but, again, also due to the fact that verification data collected up to the point of detection of the problem may be rendered invalid.

5.2.2.2 Concept development iterations

The iterative loop through a segment of the development lifecycle prior to the finalisation of an agreement for the supply of the system-of-interest was described in Chapter 4.9.3.8.

A significant situation where baselining of preceding process function outputs must be postponed until the completion of succeeding process functions exists during the stage when the system concept is developed. It is important to recognise that the agreement can only be finalised when the system architecture (including interface protocols and redundancies) is defined and when the design assurance and software levels of the constituent CIs were determined, and the extent of verification testing is known, as these are the most significant cost drivers. This implies that the product configuration data associated with the process functions i.e. Requirements Analysis, Functional Hazard Assessment, Architectural Design, Preliminary System Safety Assessment and Verification Planning, can only be baselined after all the iterations within this loop

have been completed. This approach allows for a significant reduction in the project risks in terms of costs estimation, as well as the containment of "scope creep".

The requirements should be baselined, i.e. protected against unauthorised changes, after finalisation of the agreement.

- Parallel execution of verification tests

The verification tests in the different test environments can be performed in parallel, as each of the test environments aim at demonstrating compliance against a different set of requirements. Due to the integrity of the data at the verification baseline stage, there is no need to perform verification tests in a particular sequence, e.g. rig tests, followed by ground tests, followed by flight tests, as in the typical "V" development paradigm.

5.2.3 Management of system integrity during integration flight tests

Airworthiness concepts were described in Section 4.7. When an aircraft is modified to flight test a system for which no airworthiness approval exists, the Type Certificate (TC) for that aircraft type is not valid anymore and the authorisation to operate the aircraft is revoked. In South Africa, in the case of military aircraft, authorisation to operate the aircraft for test purposes is granted by the military airworthiness authority by means of a Certificate for Flight Trials (CFT), which serves as an airworthiness approval document in lieu of a TC. The CFT is particular to a specific aircraft (identified by its tail number) and it identifies all the limitations and special instructions associated with the modification, in terms of operating the aircraft.

The assigned person responsible for the system under test prepares a Declaration of Design and Performance (DDP) document, which serves as an input to the CFT. The DDP contains special flight and maintenance instructions related to the use of the system under test and references safety assessments done to ensure the safety of the system for the intended test envelope. As part of the preparation of the DDP it must be ensured that all lifecycle data referenced in the DDP is under configuration control and that it is associated with the current status of the system under test. It is important to note that, although the verification baseline is not established at the time of the flight test, all lifecycle data relevant to the flight test, at its current status, must be managed such that its integrity is protected.

5.2.4 Fault reporting and corrective action process

Two levels of fault reporting can be discerned, i.e. the management of faults identified on baselined items, and managing faults detected on and non-baselined items.

Baselined items are managed by means of Problem Reports which are dealt with at Configuration Control Board (CCB) meetings, where changes in response to the reported problem are authorised.

Errors detected in non-baselined lifecycle data, software code and hardware are managed via an insular debugging process, which ensures that all errors are captured and addressed appropriately.

5.2.5 Process audits

Process audits need to be conducted to confirm adherence to the processes, as defined in the project plans, specifically as defined in the Plan for Software Aspects of Certification (PSAC), in the case of software, and the Plan for Hardware Aspects of Certification (PHAC), for the development of hardware.

5.3 Detailed process design

The detailed formulation of the comprehensive development process description is presented in this section. This description is a synopsis of the process as it is described in an Engineering Manual in the Denel Aviation QMS [64], and is presented to show how information deriving from the DSR relevance and rigour cycles was applied in the process design.

Note that the details presented below originate from the assessments of the engineering processes described in the literature study, as well as from real-world considerations resulting from hands-on situations encountered in practise.

5.3.1 Lifecycle model structure

The lifecycle model to be applied in the process design was derived in Chapter 4.9.3.

The system layer lifecycle model is shown in Figure 66. The lifecycle is broken down into requirements-, synthesis-, and verification groups of activities. The figure shows the flow of information: requirements are established and subjected to a functional hazard assessment, to determine the safety objectives for the system; the requirements are synthesised into a system which is subjected to verification activities, to provide evidence that the system meets with the requirements.

The system architecture is assessed during a preliminary system safety assessment, to ensure that the architecture and the assigned design assurance and software levels meet with the safety objectives. The elements constituting the architecture are specified and designed during the detailed design process function, and are

manufactured or procured during the realisation process function. Note that the detailed design and realisation process functions occurs in parallel with the development of CIs on lower layers of the hierarchy, as shown in Figure 67. The realised elements are then integrated into the system of interest. As part of the verification process, a system safety assessment is performed to provide assurance that the safety objectives, as indicated by the FHA, have been met. Figure 66 shows fundamental process functions to provide an uncomplicated view on the system lifecycle, emphasising the importance of the safety related process functions. Note that the process functions shown in Figure 66 are deconstructed further into additional process functions to improve the detailed description of the development process, in Section 5.4.

The relationships between the system layer lifecycle and the lifecycle of lower layer CIs (or subsystems) are shown in Figure 67. System layer requirements are allocated to elements of the system architecture, i.e. the CIs that constitute the final system that are identified during the system layer architectural design process function. The same generic lifecycle as used on the system layer (excluding the system safety aspects) is utilised on the lower layers of hierarchy. For each lower layer CI (hardware, software and product enabling system) the development sequence of establishing requirements, the development of subsystem architecture, detailed design, realisation, and integration, is applied. At the end of the integration process, the subsystems are delivered to the system layer, to be integrated with the rest of the subsystems to form the end product. When the integration process is completed, all the product configuration data is baselined and the system and subsystems are subjected to verification tests, analyses and reviews. An important aspect of formal tests is that these are normally witnessed by representatives from the client.

It is very important to recognise that the configuration information related to a specific build increment on all layers of the system hierarchy must be baselined, i.e. validated and protected against unauthorised changes, prior to verification. The objective of verification is to demonstrate that the realised system, described by its product configuration information, comply with the system requirements. If a non-compliance is identified during verification that needs to be corrected, the change may alter system characteristics that were verified prior to the identification of the non-compliance, rendering the relevant verification data invalid. By baselining the lifecycle data prior to verification, the process of managing traceability between requirements, synthesis and verification data is controllable.

It is also important to recognise that, due to the baselining prior to verification, verification on all layers of hierarchy can be performed simultaneously in time, e.g. software verification tests are performed along with system verification tests as the objective is to demonstrate compliance – errors should have been removed during the integration processes.

The lifecycle models for the development of software, hardware, and product enabling systems are shown in Figure 68, Figure 69 and Figure 70.

It is also important to note at this point that the outputs of each process function are validated at the level of the process function.

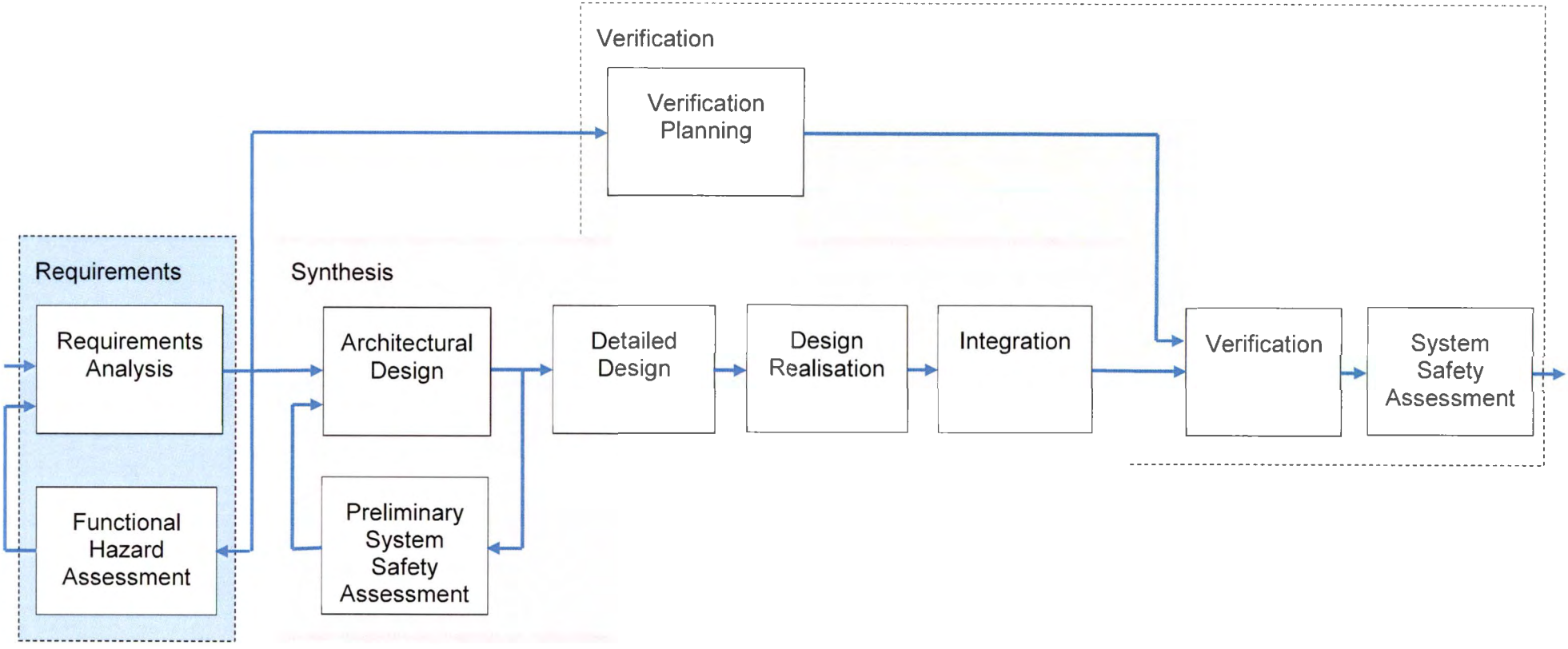

Figure 66: System layer lifecycle model

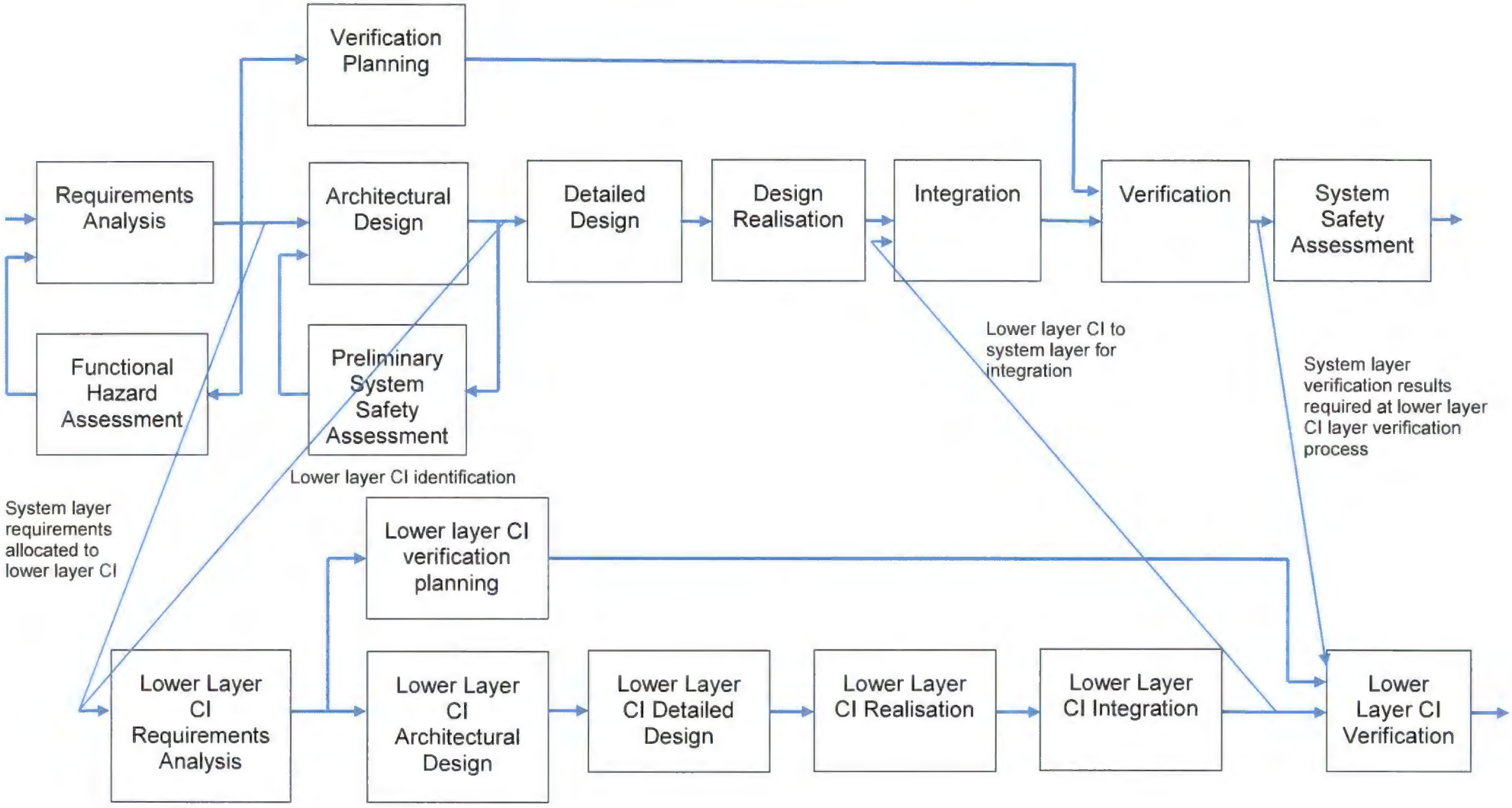

Figure 67: Interactions between system and lower layer hierarchy lifecycles

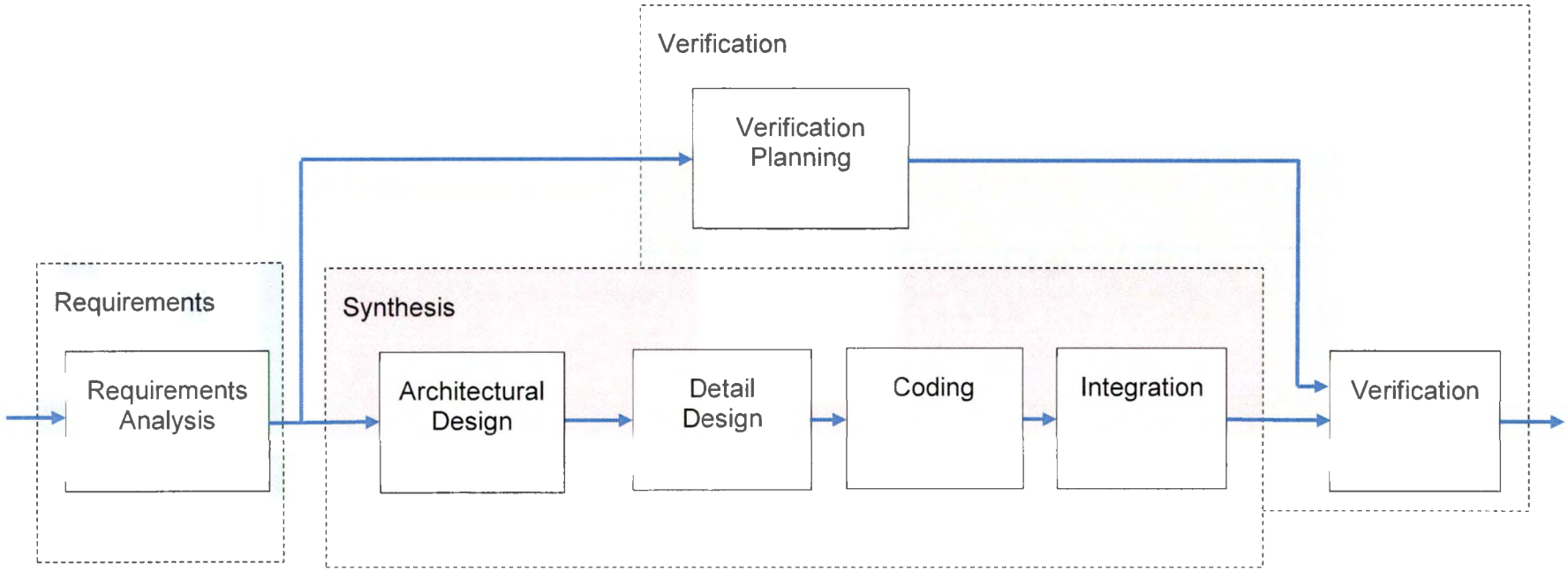

Figure 68: Software layer lifecycle model

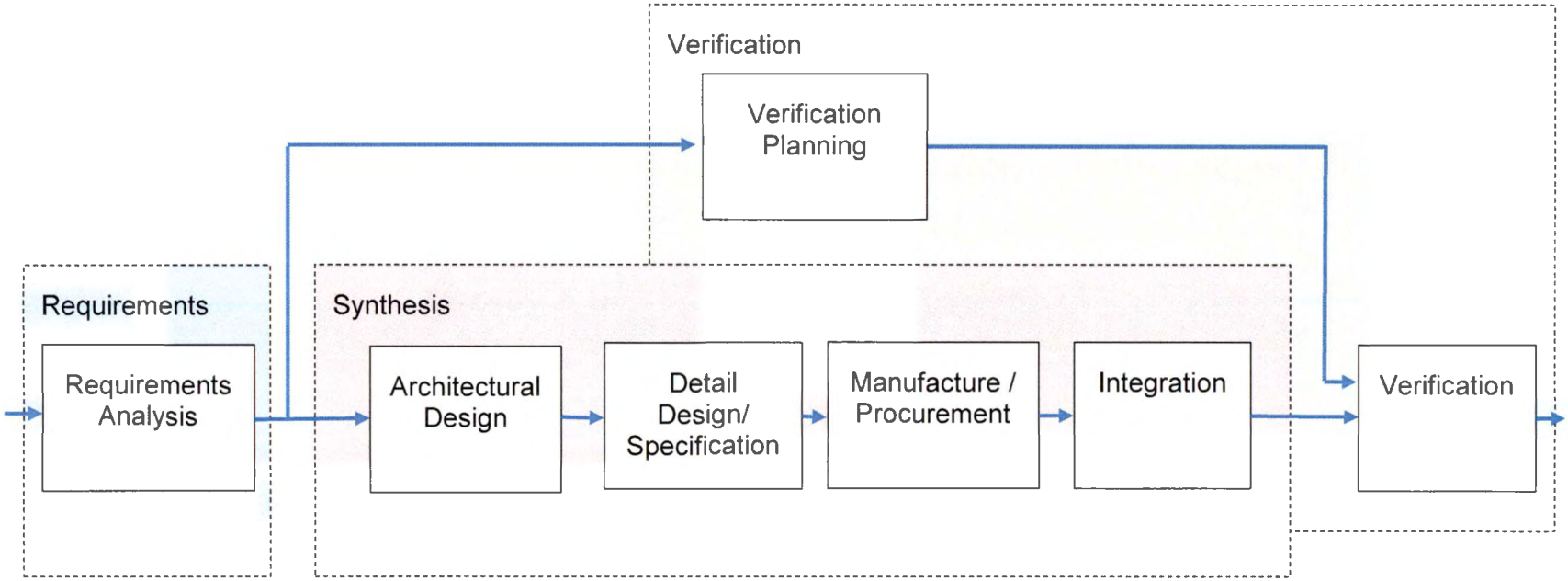

Figure 69: Hardware layer lifecycle model

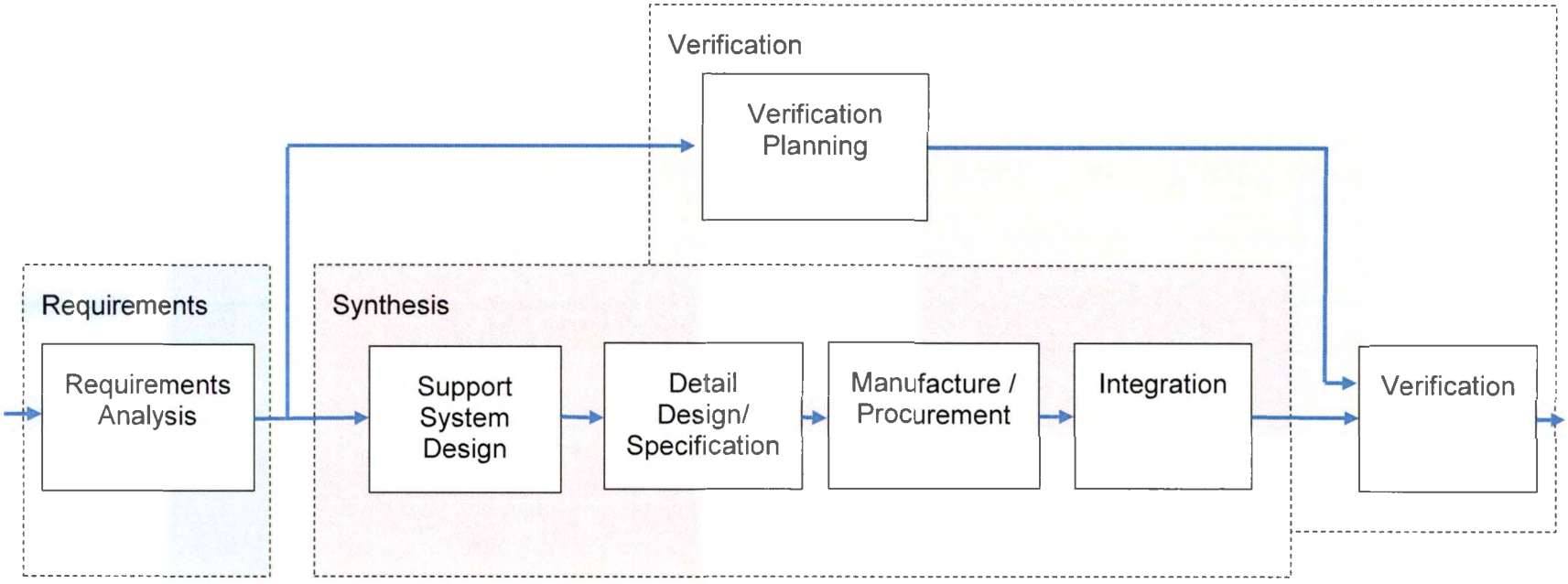

Figure 70: Product enabling systems layer lifecycle model

5.3.2 Incremental development of software functionality

The iterative application of the process model, based on the literature study, was described in Chapter 4.9.3.8.

To reiterate: the advantage of this method of delivery is that it reduces the amount of data that is to be validated and verified at a time, and it provides better system functionality to support the development of other aspects of the aircraft during development at earlier stages in the program, than what would be possible with a single linear development strategy. It also represents contractual milestones which are readily quantifiable.

The process functions affected by incremental development are shown in Figure 71.

The sets of requirements earmarked for the development "block" stages must be identified during the project planning phase.

For each block, the low level details (algorithms) for the set of software requirements to be implemented are developed, realised in source code, and integrated with the previous software build. The updated software is integrated in the target computer, and subjected to system level integration tests. When all integration problems related to the software build are resolved, the applicable lifecycle data is baselined and the system and software are subjected to verification tests.

The cycle is shown in Figure 72 for an example of three blocks.

5.3.3 Implementation of new requirements

Although the lifecycle model that was derived in Chapter 4.9.3 is in principle a pre-defined, multi-step model, it is feasible that the model can also be used as described below.

For systems employing software for its functionality, it is feasible that an existing system can be upgraded to implement additional functionality which did not form part of the original agreement. Figure 73 shows the employment of baselines to manage the implementation of new requirements with regards to an existing product baseline.

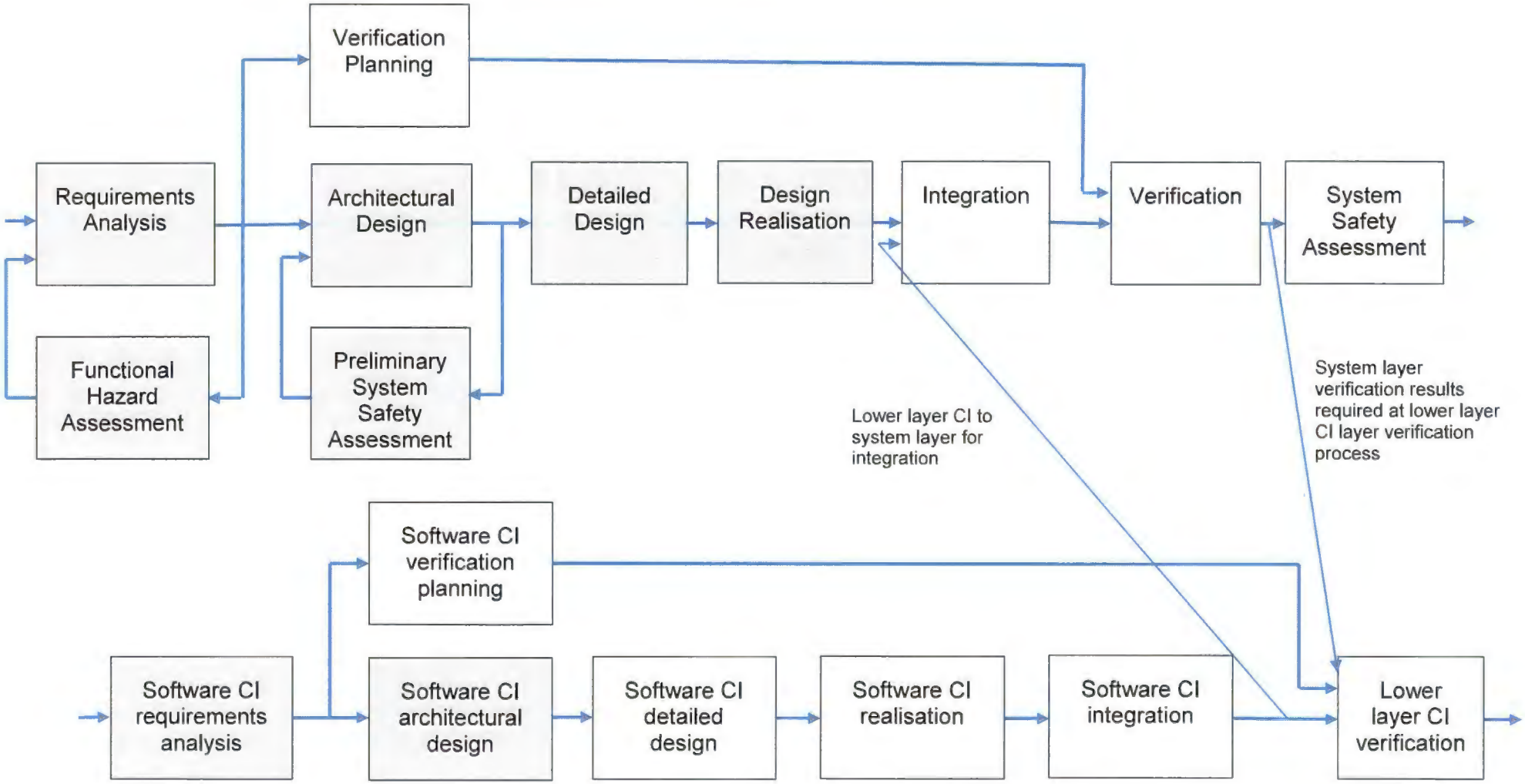

Figure 71: Incremental delivery of functionality (Software "Blocks")

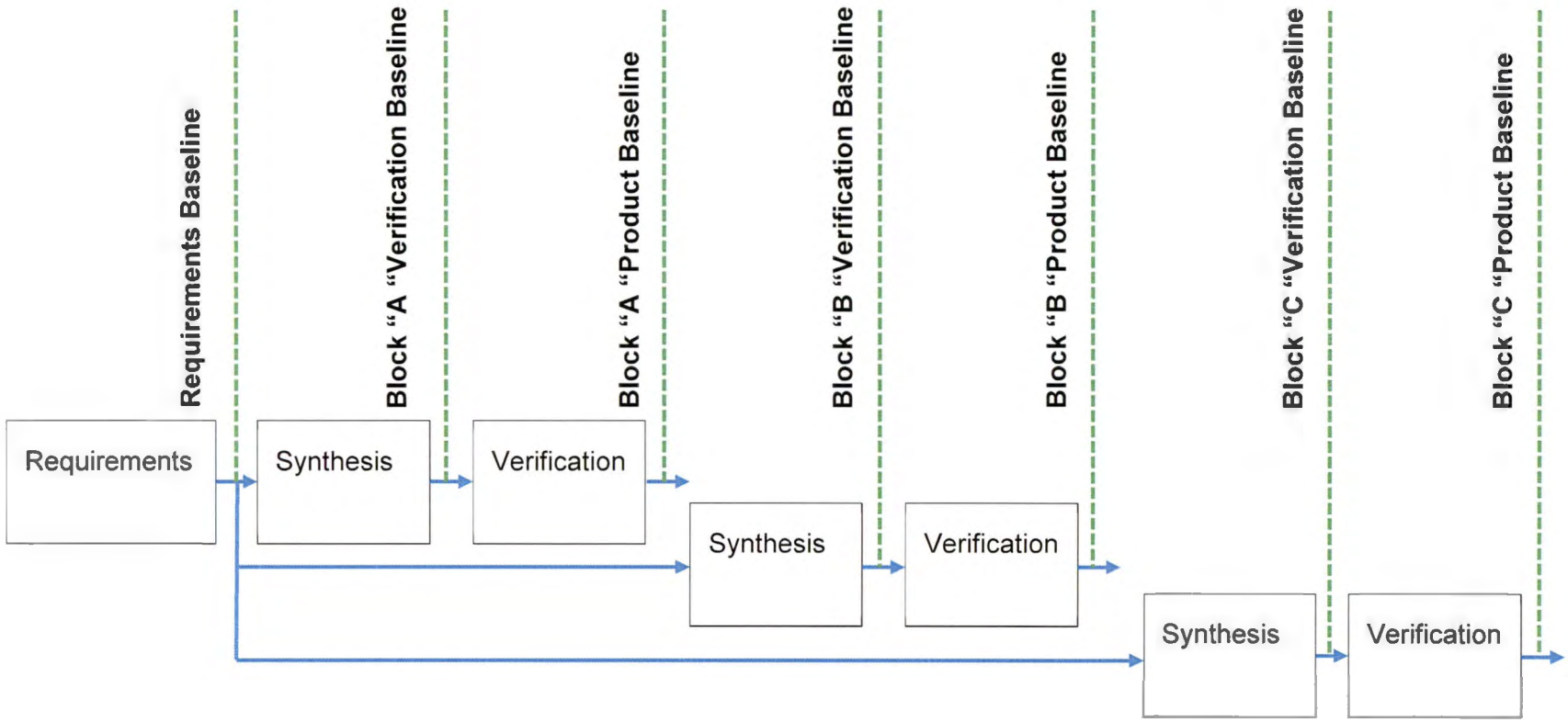

Figure 72: Incremental delivery baselines

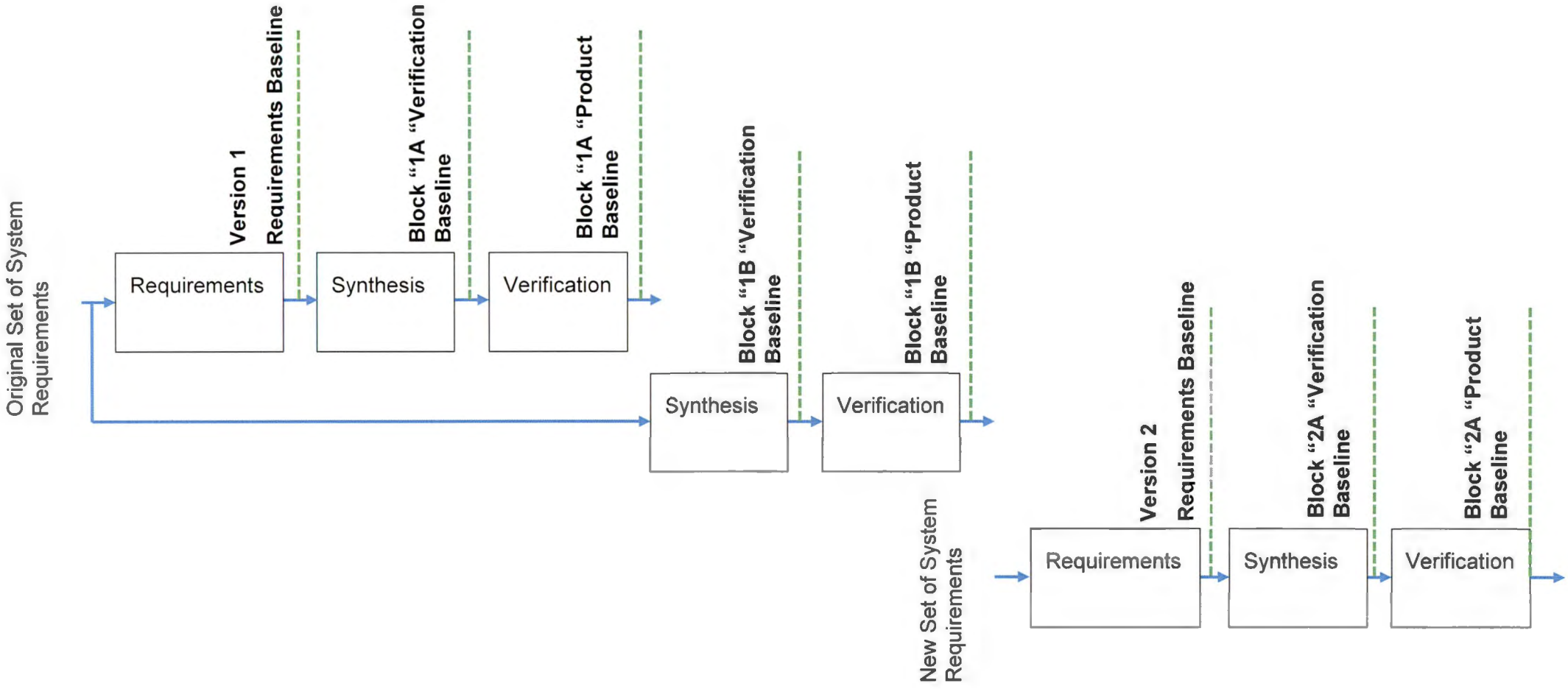

Figure 73: Implementation of new requirements (system layer)

5.4 Process function details

The process functions which constitute the development process are described below.

5.4.1 Requirements processes

The requirements processes described in this section implement principles as described in the literature study. These are summarised in Chapter 4.9.2.1.

A system is exemplified by a set of requirements and accompanying system architecture, which identify the elements (subsystems) of the system and their interrelationships (interfaces). Requirements are allocated to system elements within the system architecture and are developed on different layers of the system hierarchy. The traceability (associations) between requirements at different layers of the system hierarchy is tracked in the requirements database.

The two system layer requirements process functions are Requirements Management and the Functional Hazard Assessment, as shown in Figure 74. The system requirements are identified, recorded, analysed and allocated to system building blocks. Functional system requirements are assessed for their impact on safety and the results are fed back to the requirements management process, as safety requirements.

As discussed in chapter 4.9.3.3, requirements based verification forms the basis of the mechanism by which confidence is established in the integrity of the system, hence it is imperative that every attribute of the system is traceable to a requirement, i.e. no functionality exists within the synthesised system that cannot be associated with a stated requirement.

During the verification process, verification data affirming compliance with every requirement is acquired, and associations with this verification evidence are established in the requirements database.

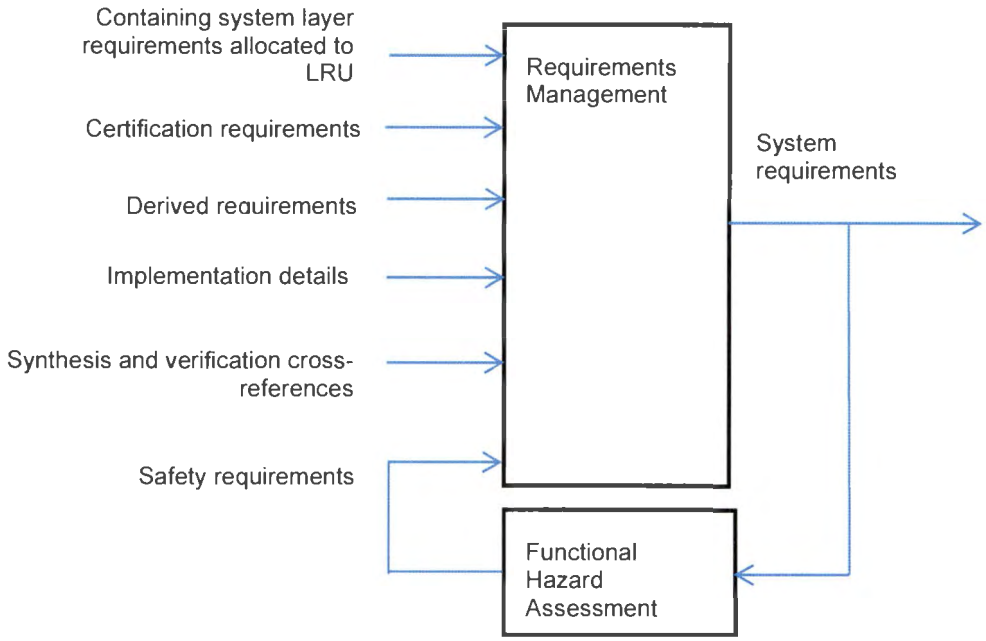

Figure 74: System layer requirements process

5.4.1.1 Requirements management

The formulation and statement of requirements are based on the concepts summarised in Chapter 4.9.2.1.

The objective of the requirements management process function is to establish system requirements and to track synthesis and verification of a system with respect to the system requirements. The requirements analysis process consists of the eliciting, formulating, refining and analysis of the system requirements, and recording of the details in a requirements database.

It must be ensured that all the sources of requirements with respect to the system-of-interest are identified, e.g. a higher layer specification, agreement, and additional airworthiness standards that may be applicable to the specific instance of the system under development. Provision must be made in the requirements database for the traceability of each requirement to an identified source. In the case of a derived requirement, it must be indicated as such in the requirements database. This reference to requirements sources supports the requirements validation process and is also informative when the database is revisited during the operational life of the system during e.g. an incident investigation or if an update is to be performed.

Requirements as contained in the identified sources must be recorded in a requirements database. A unique reference number must be allocated to each requirement to facilitate traceability and verification coverage analyses. The format of the requirements statements (e.g. the use of plain English with a defined set of grammar rules, or an identified UML subset) must be defined in the requirements standard applicable to the project.

As part of the formulation of the requirements statements, the method by which the correct implementation of the requirements, by means of the synthesis processes, are to be verified, must be determined and formulated, i.e. the verification requirements for the requirement, as well as the acceptance criteria, must be stated and recorded in the requirements database. A requirement for which the implementation cannot be verified is not a valid requirement. Note that traceability to verification evidence references should also be facilitated by the requirements database, once the verification process has been completed. Derived requirements must be identified and recorded.

The requirements must be classified with respect to requirements attributes, e.g. it must be determined whether the requirement specifies a functional, safety or performance attribute, or whether it is an implementation constraint. The types of requirement attributes, e.g. maintainability, HMI, or performance requirement type, are to be identified in the requirements standard for the project. Note that certification requirements are also managed through this methodology. If a requirement is a certification requirement, it will be assigned a "certification requirement" attribute in the requirements database, which will facilitate the tracking of certification requirements per se.

Each requirement is to be allocated uniquely to an element (or elements) of the system architecture intended to implement that requirement. This action requires interaction with the architectural design process and the PSSA. Traceability of allocated requirements to system elements and its implementation status should be facilitated by the requirements database. This will also provide for the upwards and downwards traceability of requirements.

The controls on the outputs of the requirements analysis process include the requirements validation process, i.e. the determination that the requirements are the correct requirements and that the set of requirements is complete.

When the requirements with respect to a specific build have been validated, the system specification should be developed. The system specification is a description of the

behaviour of the system to be developed and normally forms the basis of an agreement (contract). This specification is presented as a document separate from the requirements database.

The requirements process is used to track all activities related to the development of the system and remains active throughout the system lifecycle. A requirement lifecycle is associated with the implementation and verification status of each requirement. The requirements process also includes the methodology for the resolution of non-compliances. A requirement can only be closed if all discrepancies with respect to the requirement are resolved. The requirements lifecycle stages are listed in Table 16.

Table 16: Requirements Lifecycle

Lifecycle	Description
Draft	A requirement is in the "Draft" state when it is initially recorded and remains in the draft state until it is validated. The requirement statement can be modified without Configuration Control Board (CCB) authorisation while it is in the draft state.
Validated	A requirement is in the "Validated" state when the validation process w.r.t. the requirements is completed and an association with approved validation evidence or traceability to a higher level requirement or source document is verified and cross references are recorded in the requirements database. The validation process also includes confirmation that the requirement is allocated to a specific system building block or set of building blocks and that the feasibility of implementing the requirement through this building block is ratified. A set of Validated requirements is associated with a Requirements Baseline.
Implemented	A requirement is in the "Implemented" state when all integration activities regarding the implementation of the requirement are completed and all relevant design data is under configuration management. Note: A set of requirements is typically implemented together as part of an incremental build, and hence it is normal for this set of requirements to be moved to the "implemented" state together. A set of Implemented requirements is associated with a Verification Baseline.
Verified	The artefacts and lifecycle data associated with a set of requirements in the "Implemented" state is ready to be subjected to the formal verification process. A requirement is in the "Verified" state when an association with approved verification evidence is verified and cross references are recorded in the requirements database.
Closed	A requirement can only be moved to the "Closed" state if verification is completed. In cases where synthesis and verification discrepancies may have arisen these shall be resolved and an association with documentation recording the dispensation or waiver shall be verified and cross references are recorded in the requirements database. A set of Closed requirements is associated with a Product Baseline.

Changes in the status of a requirement are managed by the control process described in section 5.5.

5.4.1.2 **Functional Hazard Assessment (FHA)**

As explained in Chapter 4.7, the FHA forms the system safety part of the requirements management process.

An FHA must be performed on functional requirements details as formulated in the requirements database to determine the severity of the hazards posed by functional failures. The FHA identifies safety requirements to be imposed on the design and development processes of the system. The FHA further assesses the consequences of functional failures and categorises identified failures in terms of the severity of the consequence of the failure, i.e. according to the hazards resulting from the failures. An FHA consists of studies by individuals as well as discussions in workgroups.

The resulting safety requirements must be recorded in the requirements database. Any new functional requirement that has been identified or added during the system lifecycle shall be subjected to the FHA process.

Changed functionality, for example, where GPS data had originally been used for navigation purposes but where the time data derived by the GPS were used by the secure communications for time synchronisation purposes in due course during the system lifecycle, will be subjected to the FHA process.

Hazards raised by the hazard process on higher layers of hierarchy, allocated to the system of interest, shall be subjected to the FHA process.

5.4.1.3 **Lower layer requirements processes**

Lower layer processes implement the same generic lifecycle as for the system-of-interest layer. This implements the principles described in Chapter 4.9.1.2.

Only the requirements management process function is defined on the subsystem layers as a requirements type activity, as shown in Figure 75, i.e. the FHA is only performed at the system layer. The subsystem (hardware, software and product support system) requirements management process is conceptually similar to the system layer requirements management process, i.e. requirements are identified, recorded, analysed and allocated to subsystem building blocks.

This requirements process also remains active throughout the subsystem lifecycle and is used to identify the subsystem requirements and manage the technical progression of the implementation and verification of the subsystem requirements.

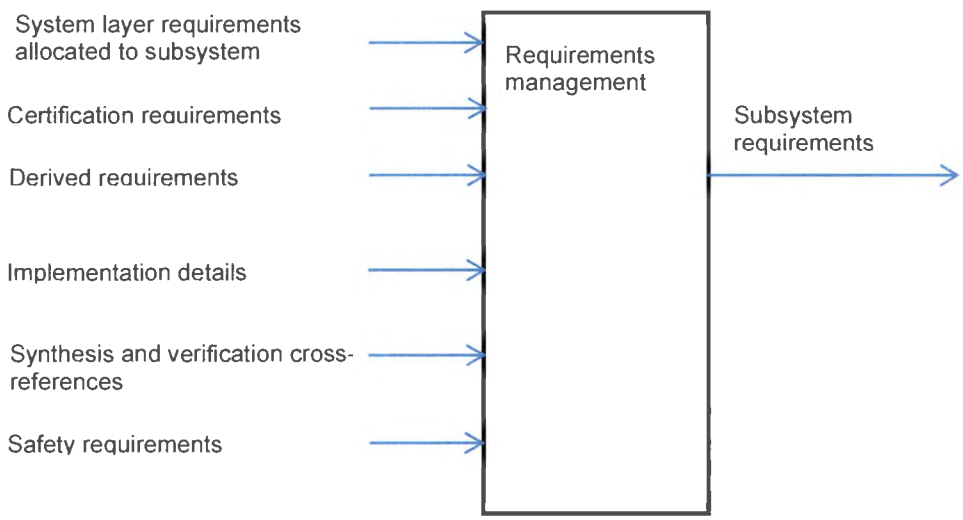

Figure 75: Subsystem layer requirements process

The lifecycle data developed by the requirements processes are listed in Table 17.

Table 17: Requirements process lifecycle data

No	Item	Objective	Reference	Applicability	Notes
1	Operational concept definition	Describes a proposed system in terms of the user needs it will fulfil, its relationship to existing systems or procedures, and the ways it will be used.	MIL-STD-498 DID DI-IPSC-81430	AV system Product	The OCD is used to obtain consensus among the acquirer, developer, support, and user agencies on the operational concept of a proposed system
2	Functional Hazard Assessment	Identifies the hazards associated with functional failures	SAE ARP4761	AV system Product	1. Not considering implementation methods or mitigation. 2.Extracted from requirements database at given baseline level
3	Specification tree	Defines the elements and relationships w.r.t project specifications	N/A	AV system Product Hardware	Required to index lifecycle data
4	System/subsystem specification	Describes the requirements and verification of the requirements for a combination of elements that must function together to produce the capabilities required to fulfil a mission need, including hardware, equipment, software, or any combination thereof.	MIL-STD-961E	AV system Product Hardware	Extracted from requirements database at given baseline level
5	Software specification	Describes the requirements and verification of requirements for the automatic acquisition, storage, manipulation, management, movement, control, display, switching, interchange, transmission, or reception of data or information.	MIL-STD-961E	Software	Extracted from requirements database at given baseline level

No	Item	Objective	Reference	Applicability	Notes
6	Requirements resolution item	Manages resolution of requirements anomalies	N/A	AV system Product Hardware	Replaces CRIs
7	Declaration of design and performance	Formally state compliance with specifications, list non compliances and address anomalies	N/A	AV system Product	Airworthiness declaration document
8	Engineering report	Record results of a technical investigation or analysis	N/A	All	To be used for all types of engineering reports which are used for verification evidence or which are to be distributed to external parties

5.4.2 Synthesis processes

The synthesis processes described below implement concepts stemming the literature study, as summarised in Chapter 4.9.2.

The system layer synthesis process functions are shown schematically in Figure 76. The synthesis processes realise the current incremental build of the system of interest as defined by its set of requirements. The system layer synthesis process functions, as shown in Figure 76, are:

- Architectural Design;
- Preliminary System Safety Assessment (PSSA);
- HMI Design;
- Detailed Design;
- Procurement of Subsystems, Software and Services;
- Bench Integration;
- Air Vehicle Integration;
- Integration Flight Test and
- Preparation of the Verification Baseline.

The system architecture is developed in accordance with system requirements. The architectural design is analysed by the PSSA to ensure that the architecture satisfies safety requirements as established by the FHA and to determine software and design assurance levels required to meet with safety requirements. System layer HMI and detail design aspects are developed and items for procurement are specified. Building blocks from the subsystem and software development layers and other procured subsystems are then integrated in an appropriate integration environment to produce a specimen (or sets of specimens) which can be subjected to integration flight tests, if required. Before the system can be cleared for flight, it must be shown to meet with a set of minimum environmental requirements, and is hence subjected to the Safety of Flight (SoF) environmental tests, prior to the commencement of the integration flight tests. When the integrity of the system on the aircraft has been ensured (i.e. the experimental installation does not pose a safety risk), the system is subjected to flight testing to support integration objectives. At the point in time where the system is ready

to undergo a series of verification tests, all lifecycle data and test specimens are finalised and its configuration integrity is confirmed to ensure a stable baseline with which the verification data can be associated.

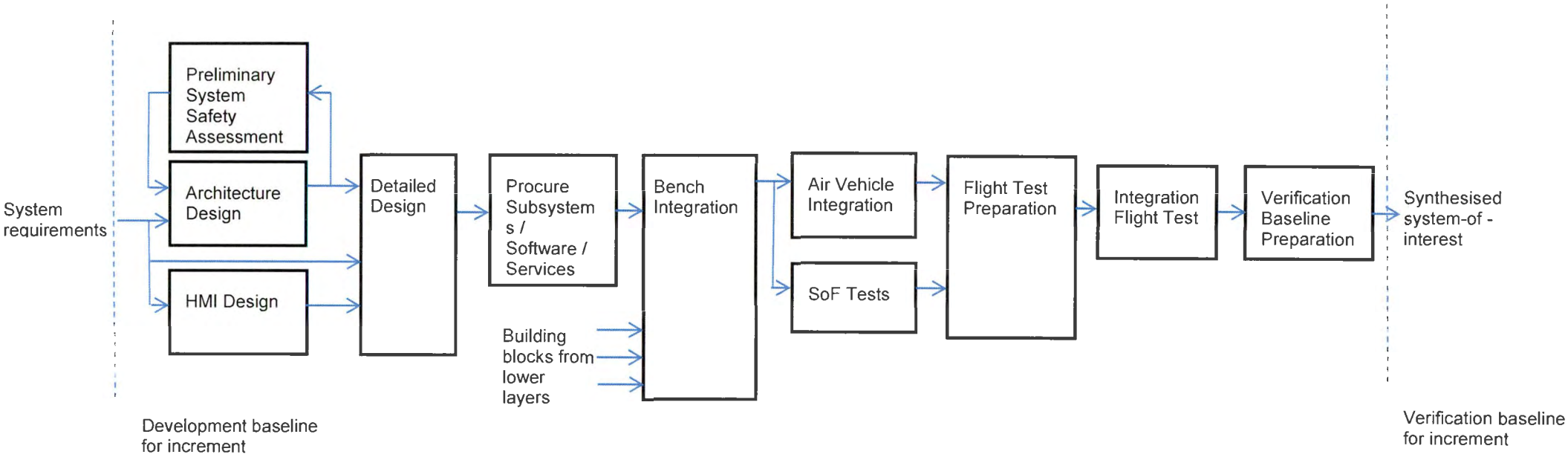

Figure 76: System layer synthesis process functions

5.4.2.1 Architecture design

The architecture design process function implements the high-level design activity, described in Chapter 4.9.2.2.

The objective of this process function is to develop a system architecture that is consistent with the requirements defined during the requirements processes.

The system architecture design includes the detailing of subsystem and interface particulars. The architecture contains definitions of building blocks to which requirements are uniquely allocated. The architectural design must be consistent with safety requirements, e.g. if a failure mode of a single element of the system architecture can contribute to a catastrophic failure, then the probability of that failure shall be less than 10^{-9}. If that cannot be achieved, then the architecture must be adapted to incorporate sufficient redundancy to achieve the safety requirements.

Architectural design trade-off studies must be performed and the results should be recorded in engineering reports. These reports serve to provide traceability of engineering decisions.

An important element of the design of avionics is the determination of the appropriate data interface protocols, e.g. ARINC 429, MIL-STD-1553B and RS-422. The selection of data interface protocols must be done prior to the finalisation of development agreements, as these have a significant impact on the system lifecycle costs. The interface protocol selection criteria must be identified and recorded to record design decision rationale and to improve traceability. Interface details shall be recorded and maintained in a system layer Interface Control Document (ICD).

The system architectural design details must be provided to the system layer requirements process in order to allocate requirements to subsystem elements.

The system architecture provides a basis for the Product Breakdown Structure (PBS) which in turn leads to a Work Breakdown Structure (WBS) for use by the project management team.

It was found that the pro forma for a number of project documents, e.g. plans, specifies a system description section, which describes the system under development, to improve the readability of the specific document. This process of producing a system description for each document may lead to discrepancies if the editing of the description is not carefully managed. As the high level system design is an implicit outcome of this process function (architectural design), a high level System Description is developed to summarise the architectural design

details in a format that can be used or referenced by other project documents, eliminating the possibility of discrepancies in terms of the system description between project documents.

5.4.2.2 Preliminary system safety assessment

The integration of the PSSA with the development process is described in Chapter 4.9.3.4.

The objective of the Preliminary System Safety Assessment (PSSA) is to analyse the system architecture to ensure that the safety requirements will be satisfied.

Design features required to ensure that the system meets with the safety requirements are determined. These typically involve the specification of redundant elements and determination of design and software assurance levels. PSSA methods include Fault Tree Analyses (FTA) and Failure Mode and Effect Analyses (FMEA). The PSSA consists of studies by individuals, as well as discussions in workgroups.

As part of this process function, it must be ensured that the findings of the PSSA are communicated to all stakeholders and that the assurance level requirements are addressed in downstream engineering activities.

The PSSA outcomes (design and software assurance levels) have a significant impact on the project costs. This implies that the PSSA must be performed prior to the finalisation of development agreements.

5.4.2.3 Human-machine interface design

Due to the prominence of HMI aspects in the type of system described here, and the importance of involvement of potential users and its impact on flight safety, the HMI design is dealt with as a topic on its own in the design of the process.

The objective of this process function is to design system level human-machine interface (HMI) details. The system layer HMI design is developed to be commensurate with the HMI requirements and applicable HMI standards, and recorded in the HMI Design Description document. HMI aspects include the design of controls, displays, illumination, auditory elements, etc.

Note that the HMI requirements, i.e. the HMI functions to be performed, are developed as part of the Requirements Management process function.

The HMI design will eventually be described in the section of the flight manual which describes the operation of the system (the development of the flight manual entries form part of the

product support process development), but for flight test purposes, while the manuals are not published yet, HMI details are presented in a "Pilot's Information Leaflet".

The HMI design process involves the participation of the operator of the system, and their inputs are captured in the minutes of HMI workgroup meetings.

The system layer HMI document may be used as a software design document where applicable.

The data in the Pilot's Information Leaflet should be developed during the integration processes, using aircrew inputs, and eventually be incorporated in the flight manual Section dealing with the specific system.

5.4.2.4 Detailed design

The detailed design process function implements the low-level design activities described in Chapter 4.9.2.3.

The objective of this process function is to develop the detailed design at the system layer. This process function entails all the details affecting the design of the system, e.g. reliability, availability and maintainability (RAM), electromagnetic compatibility and producibility.

Design details are captured in detailed design documentation (drawings, CAD files, etc.). The System Detailed Design Description describes details that are required to manufacture the system, which are not contained in the architectural design or at lower layer design details.

The Detailed Design Description document is also used as a document to support the verification process, to provide evidence where the means of compliance is "Review".

The details for the items and services to be procured that were identified during the architectural design process are determined and specified as part of the detailed design process.

An important aspect of the detailed design process function is to ensure that derived requirements originating during the detail design process is provided to the requirements processes, as explained in Section 4.7.3.1. This activity requires interaction between the persons responsible for requirements management and the design specialists.

The detailed design process typically includes engineering studies to assess design trade-offs, size components, parameter ranges, etc. In order to provide traceability, the outcomes of these studies should be captured in engineering reports.

5.4.2.5 Procurement

The procurement process function is inserted here in the synthesis lifecycle as a practical measure, to facilitate planning and control.

The objective of this process function is to interface with the supply chain with respect to subsystems and services identified for procurement at the system development layer. The procurement process is initiated from a technical processes perspective.

This process function includes sub-contracting. The sub-contract management is performed by the project processes. The technical processes must ensure the integrity of the information provided to the project processes.

Note that the procurement described here typically does not involve the procurement of production volumes. However, when procuring systems for development purposes, care should be taken that the supplier will be able to meet with production and lifecycle support requirements.

5.4.2.6 Bench integration

The integration activities form a critical component of the synthesis process. Practical experience has shown that integration problems can have severe effects of project schedules and costs. The integration processes described in the subject matter literature are summarised in Chapter 4.9.2.5.

The objective of this process function is to integrate all the system components to produce the system of interest, ready for verification. Subsystems developed at lower layers or procured from suppliers and sub-contractors are integrated in an appropriate integration environment. This environment will normally also be used for verification testing as well. The development and management of this environment is described in Section 5.6.

The insular debugging process should be used to record and track problems for feedback to upstream development activities.

It is important to note that the integration tasks include a significant amount of testing. Most of these integration tests may need to be repeated during verification once integration is completed. Integration tests may also identify errors in the requirements specifications. In a system-of-systems view, significant loops (data flow) exist between requirements and integration processes on the different layers of hierarchy.

The integration process should include the execution of "dry runs" of the test procedures to confirm that the test environment is sufficient and the system under test is, as well as to

remove errors from the test documentation. These "dry runs" can be used for validation of the test specifications.

As the verification baseline is not established during integration (as designs may change to resolve integration problems), the integration test results cannot be used as verification evidence.

5.4.2.7 Air vehicle integration

Air vehicle integration is described as a separate entity in this process formulation, as stricter control measures are applied in terms of aircraft modifications, as compared to bench integration.

The objective of this process function is to integrate a pre-verification version of the system under development in the test aircraft for integration flight testing purposes. The system under development is installed and integrated on the test aircraft. Aircraft integration procedures and practises must be adhered to when performing integration modifications.

Before the system is cleared for flight to test a development aspect, a formal ground test procedure (Integration Phase Ground Test Procedure) must be performed to establish that the functionality to be tested is of a sufficient level of maturity and to confirm that all safety aspects are controlled. The resulting test report forms part of the documentation required to clear the aircraft for flight.

The integration process should include the execution of "dry runs" of the test procedures, to confirm the maturity of the integrated system, as well as to remove errors from the test documentation. These "dry runs" can be used for validation of the test specifications.

As in the case of bench integration tests, a verification baseline is not established during integration and the integration test results cannot be used as verification evidence.

5.4.2.8 Safety of flight environmental testing

The process function described in this section is incorporated into the synthesis process due to practical considerations.

The objective of this process function is to confirm that the system under development meets with environmental criteria to allow the installation on the test aircraft for integration flight test purposes and that it will not pose a safety hazard.

These tests are pre-requisites for integration flight tests and can be performed in parallel with the air vehicle integration process function. These tests should be done to such an extent that shortcomings (vibration, electromagnetic interference (EMI) or Electromagnetic Compatibility (EMC), etc.) can be identified and rectified prior to the environmental verification tests where the compliance to environmental tests are demonstrated.

The safety-of-flight tests are generally sub-contracted to specialist companies.

5.4.2.9 Flight test preparation

Flight testing at Denel Aviation is performed by the Flight Test group, who have their own policies and procedures defined in the Denel Aviation QMS. This process function is defined to provide a method to co-ordinate communication and activities between the engineering team and the flight test team.

The objective of this process function is to develop integration phase flight test requests and to ensure that all the documentation required for the Certificate for Flight Trials (CFT) (see Section 5.2.3) meet with set criteria. The configuration status of the system under test for each flight test is accomplished by this activity.

Flight test requests, indicating integration Flight Test Requirements (FTRs), are to be developed. The FTRs include descriptions of e.g. flight profiles to be executed as well as data requirements. The FTRs do not have to be subjected to the normal lifecycle data validation process. The flight test request is an input to the Flight Test Plan. The FTR is implicitly validated during the development of the flight test plan. The formal reference documents will be the Flight Test Plan and the Flight Test Instrumentation (FTI) design descriptions, which are referenced in the CFT.

The formal document which provides safety related inputs to the CFT is the system layer DDP. As part of this process function, the integration phase DDP for the system under test is prepared and approved. This is to be done every time the configuration of the system under test is changed from the previous flight test.

The integration phase flight tests should be used to ensure that all potential non-compliances are resolved prior to verification testing. As in the case of integration rig and ground tests, integration flight test results cannot be used as verification evidence.

5.4.2.10 Integration flight tests

This process function is also specified as part of the synthesis process to co-ordinate communication and activities between the engineering team and the flight test team.

In the particular instance of airborne equipment, integration testing may require flight tests. The objective of this process function is to perform flight tests to support the system integration process. The integration stage flight tests are performed as per the flight test plan (under the control of the flight test organisation).

As described in the literature study, airworthiness is managed through design, manufacturing process control, maintenance actions and specifications for the operation of the aircraft, the latter being managed by means of flight manuals.

At Denel Aviation, the flight test program is executed under the auspices of the Flight Test Group (FTG). Engineering provides the FTG with flight test requests, which specify flight test requirements. The flight tests are designed, planned and executed by the FTG. Activities such as design and development of flight test instrumentation are performed by the FTG.

The FTG provides flight test data and flight test reports to engineering. Typically, the test data is analysed by engineering, who provides subject matter expertise and assessment where applicable. The results of these assessments are captured in engineering reports.

The insular debugging process is used to record problems identified during integration testing for feedback to upstream development activities.

Integration flight tests are not used for requirements verification purposes; hence flight test reporting is less formal and is aimed at resolving integration problems.

Flight test reports are not subjected to change control. A test report cannot be updated. If a test is repeated, a new test report with a different configuration number is generated.

5.4.2.11 Verification baseline preparation

This process function is also specified on the basis of practical experience, to provide a mechanism for the co-ordination of the preparation of all life cycle data required for verification process.

The objective of this process function is to ensure that all lifecycle data meet with the criteria for the establishment of a baseline against which the system is verified and to then index this data. This process function is used to ensure that all the prerequisites for formal verification of the implementation of a predetermined set of requirements are met.

A configuration audit on all lifecycle data associated with the current build must be performed and the product Master Record Index (MRI) must be updated to reflect the status of all lifecycle data associated with the current build of the system.

It must further be ensured that the requirements database reflects the implementation status of the requirements for which the implementation is to be verified, the status of all affected requirements must be updated to "implemented".

The product verification environment configuration index must be updated. This index lists all the tools and version numbers thereof that will be used to perform verification tests at the system layer of hierarchy, e.g. the simulation software used in the test bench.

The verification phase DDP for the system under test (required for verification phase CFT) must be prepared.

5.4.2.12 Subsystem synthesis processes

The subsystem layer synthesis processes are shown schematically in Figure 77. The subsystem synthesis processes follow the same generic pattern as the system layer synthesis process. As illustrated in Figure 77, the synthesis process functions are:

- Architectural Design;
- Detailed Design;
- Procurement of Components and Services;
- Realisation of Subsystem Elements;
- Subsystem Integration and hand-over to system layer integration;
- The preparation of the subsystem layer verification baseline.

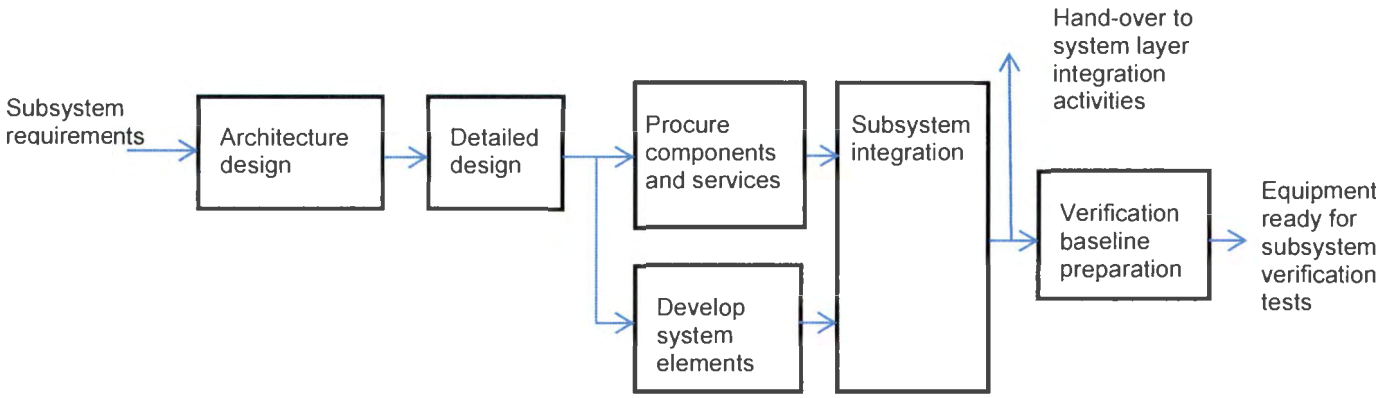

Figure 77: Subsystem layer synthesis process functions

The lifecycle data to be produced during the synthesis processes are listed in Table 18.

Table 18: Synthesis process lifecycle data

No	Item	Objective	Reference	Applicability	Notes
1	System architectural definition	Defines areas of solution expressed as a set of separate problems of manageable, conceptual and, ultimately, realizable proportions	ISO/IEC 15288:2008 Section 5.5.4	AV system Product	Contains definition of system elements and interrelationships
2	System description	Describes the system in general terms.		Product	Common reference for general purposes
3	System detailed design description	Describes detailed elements of the system.		AV system Product	Used to support MoC review
4	Human-machine interface description	Describes HMI operation		Product	Communicates HMI design with user, software developers
5	Top-Level Drawing	Identifies the hardware item and identifies all assemblies, subassemblies, components and relevant documentation that define the hardware item	RTCA/DO-254 Section 10.3.2.2.1	Hardware	
6	Assembly Drawings	Include additional detailed information needed to assemble the hardware item, assembly, or subassembly	RTCA/DO-254 Section 10.3.2.2.2	Hardware	
7	Installation Control Drawings	Ensure correct installation of a hardware item into a system or correct installation of a hardware item into another hardware item	RTCA/DO-254 Section 10.3.2.2.3	Hardware	For some lower level hardware item, assembly drawings for the next higher hardware item or assembly may act as the installation control drawing

No	Item	Objective	Reference	Applicability	Notes
8	Hardware/Software Interface Data	Identifies data relating to the interface between the hardware and the software	RTCA/DO-254 Section 10.3.2.2.4	Hardware	The performance of the hardware as determined by the requirements specification may depend upon the configuration of the hardware by the software, calibration of the hardware by the software or upon a necessary interaction between the hardware and software
9	Interface control document	Manages interface details between architectural elements		AV system	
10	Preliminary system safety assessment	Identifies and classifies hazards associated with functional failures	SAE ARP4761	AV system Product	Supports architectural design and motivates detailed design constraints
11	Software architectural design description	Defined main software modules and interrelationships, methods of execution	RTCA/DO-178C Section 11.10	Software	178 calls for a design description including architecture and detailed design. We split these as the architecture must be known for PSSA purposes, e.g. partitioning aspects.
12	Software detailed design description	Describes detailed design of software modules, e.g. algorithm details	RTCA/DO-178C Section 11.10	Software	
13	Software configuration index	Identifies the configuration of the software product	RTCA/DO-178C Section 11.16	Software	The SCI can contain one data item or a set (hierarchy) of data items. The SCI can contain the items listed below or it may reference another SCI or other configuration identified data that specifies the individual items and their versions

No	Item	Objective	Reference	Applicability	Notes
14	PSS Definition document	Define product support system elements		PSS	
15	Engineering report	Record results of a technical investigation or analysis	N/A	All	To be used for all types of engineering reports which are used for verification evidence or which are to be distributed to external parties

5.4.3 Verification processes

A summary of verification processes as they are defined in relevant standards and recommended practises was presented in Chapter 4.9.2.6.

The verification process flow is shown in Figure 78. Development of the system verification specification commences in parallel with the synthesis processes. When all the requirements associated with the current incremental build is at "implemented" status, the system is subjected to tests to verify that the system meets with the requirements. Additional confidence in the correctness of the test procedures will be established if the test procedures were subjected to "dry runs" during the integration phase.

It is important to note that, due to the establishment of a verification baseline, all classes of verification tests, as shown in Figure 78, are performed in parallel. The test reports resulting from the verification tests are reviewed by the normal validation process. The review results provide confidence that the test results are correct to achieve the RTCA/DO-178C requirement from Section 6.4.5.c of the standard (i.e. test results are correct and discrepancies explained). The test results are assessed by the system safety assessment to corroborate claims that the system is safe to operate. After the completion of all verification activities, all the verification data, as well the system safety assessment, are assessed for correctness and completeness and a process is followed to ensure that all discrepancies are resolved.

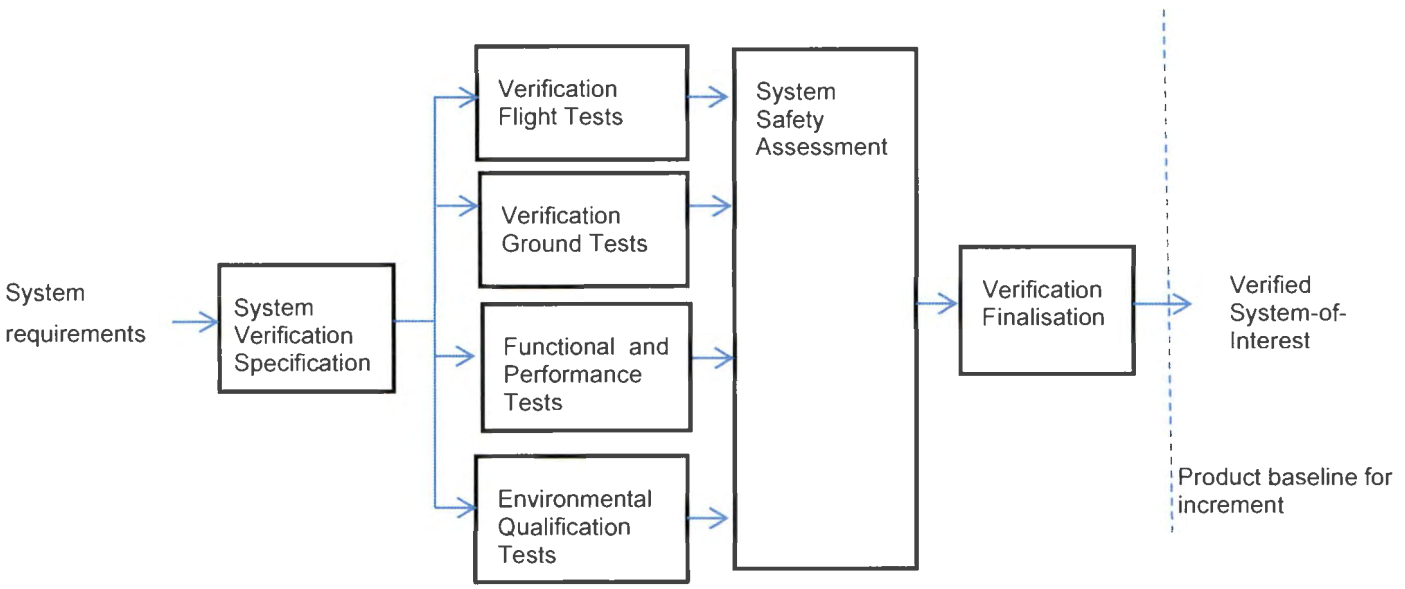

Figure 78: System layer verification process

5.4.3.1 System verification specification

The objective of this process function is to develop the verification cases and procedures that will be used to demonstrate that the system complies with its requirements. These requirements verification specification development activities should be conducted in parallel with the system synthesis processes.

The system verification requirements captured in the requirements database identify verification requirements for every requirement, including environmental test requirements. These verification requirements as well as implementation details from the synthesis processes are used to develop the test specifications (test cases and procedures).

The System Architectural Design Description is used to design tests to show that the system architecture complies with the requirements, e.g. handling of failure cases. The Interface Control Document is used to design tests to show that interface requirements are satisfied, as well as to design test instrumentation to acquire test data from these interfaces. The HMI Design Description is used to design tests using test inputs via the HMI devices. The System Design Description is used to design tests to show that the system design meets with the requirements.

The Means of Compliance (MoC) for each requirement should be identified and be agreed upon with the customer. The appropriate MoC is to be selected from the following:

Review: This method entails reviews of lifecycle data. The outcome of this activity is a qualitative assessment of correctness. This MoC is used when analyses or tests are not feasible or impractical and to provide evidence of compliance, where applicable. Reviews can also entail visual inspections on a representative sample of the system under development where success is determined by observation alone, e.g. quantitative measurements such as dimensions or other observations which do not require the system to be powered up.

Analysis: Analyses use established technical or mathematical models or simulations, algorithms, charts, graphs, circuit diagrams, or other scientific principles and procedures to provide evidence that stated requirements were met (MIL-STD-961 E). An analysis provides repeatable evidence of correctness.

Rig test: This MoC entails testing on the system, as installed onto integration and test rigs, against test procedures.

Ground test: This entails tests on the system as installed on the air vehicle, against test procedures. These tests do not include tests where it is required to start the engines.

Flight test: This entails tests on the system as installed on the air vehicle, where a flight crew will be required to perform the tests. These tests include all tests where it is required to start the engines, i.e. even if the intention is not to take off.

The following classes of test specifications are to be developed:

Environmental Qualification Test Specifications identify the RTCA/DO-160 type tests required to qualify the system for use, and it details the categories of the standard applicable to the system under test.

Verification Rig Test Procedures define the test cases and procedures to verify compliance with system requirements, using e.g. the integration bench.

Verification Air Vehicle Ground Test Procedures define the test cases and procedures to verify compliance with system requirements, using the system as integrated on the air vehicle.

Verification Flight Test Requests specify the flight tests required to verify compliance where flight tests were identified as the MoC.

This process function also provides inputs to the development environment establishment and support process.

5.4.3.2 Verification ground tests

The objective of this process function is to demonstrate the compliance of the synthesised system with the requirements where the MoC was specified to be Ground Tests, to demonstrate functional and performance achievement of requirements, with the system installed and integrated into the air vehicle. A representative specimen of the integrated system, at the appropriate verification baseline status and integrated onto the aircraft, is to be subjected to verification tests. Note again that ground tests denote tests on the air vehicle, where the system is not powered from the on board power generation system, except for tests using battery power. Tests requiring engine power are classed as flight tests.

The tests identified in the ground test procedures are to be executed. These tests are witnessed by an internal (Denel Aviation) Quality Control Representative (QAR) as well as by a QAR nominated by the customer. The test results are captured in the test report associated with the ground test procedures.

5.4.3.3 Verification flight tests

The objective of this process function is to demonstrate the compliance of the synthesised system with the requirements where the MoC was specified to be flight tests, to demonstrate functional and performance achievement of requirements, with the system installed and integrated into the air vehicle.

The flight test program is executed under the auspices of the Flight Test Group (FTG). Engineering provides the FTG with flight test requests that specify flight test requirements. The flight tests are designed, planned and executed by the FTG.

Test personnel should be made aware of the specific requirements for which statements regarding compliance are required.

The FTG provides flight test data and flight test reports to engineering. Typically, the test data is analysed by engineering who provides subject matter expertise and assessment where applicable. The results of these assessments are captured in engineering reports.

The flight test plan will be prepared by the flight test organisation, in response to the flight test request (FTR). The FTR is the engineering document against which flight test results should be produced, and should be associated with specific requirements and verification requirements. The FTR is implicitly validated during development of the flight test plan. The formal reference document is the Flight Test Plan, and the Flight Test Plan shall be traceable to the FTRs and hence to system requirements.

Flight safety is managed through the CFT process, which is managed by the flight safety department.

5.4.3.4 Environmental qualification tests

The objective of this process function is to demonstrate the compliance of the synthesised system with the environmental requirements. The environmental qualification tests are performed as specified in the test specifications and the applicable standard, e.g. RTCA/DO-160F.

These tests are typically performed in parallel with functional and performance verification tests, as they must be performed on a test specimen representative of the final product (for the development increment) and are normally subcontracted to specialist companies.

5.4.3.5 Functional and performance verification tests

The objective of this process function is to demonstrate the compliance of the synthesised system with the requirements where the MoC was specified to be Rig Tests to demonstrate functional and performance achievement of requirements. The tests identified in the rig test procedures are executed. These tests are witnessed by an internal (Denel Aviation) QAR as well as by a QAR nominated by the customer. The test results are captured in the test report associated with the rig test procedures.

Rig tests include all tests performed in a test setup that emulates the target environment in some or other manner. Rig tests complement flight tests, and are performed to obtain test data where flight testing is impractical or not cost-effective. It shall not be a pre-requisite to complete rig tests prior to commencement of verification flight tests.

In all cases where rig test results are presented as verification evidence, the rig should be subjected to a verification process. This implies that the rig should be treated as a system on its own and be subjected to the generic system development process comprising requirements, synthesis and verification lifecycles. The test setup shall be under configuration management.

5.4.3.6 System Safety Assessment

The association of the SSA with the verification process was discussed in Chapter 4.9.3.4.

The objective of this process function is to verify that the system, as designed and implemented, meets with the safety requirements, e.g. specification of redundant elements.

The System Safety Assessment is to be conducted as per ARP4761. The resulting SSA Report provides assurance that the system meets with the safety requirements. Engineering reports (FMEA, etc.) are produced to capture analysis results that were used in the process of developing the SSA.

5.4.3.7 Verification finalisation

This process function is defined in order to centralise the steps where the verification process outcomes are consolidated.

After completion of the verification processes the requirements in the requirements database are closed and cross references to verification evidence is established. The System Layer Verification Report and the final DDP for the development increment are issued.

All requirements in the database are to be closed and cross references to verification data should be recorded, using the requirements management system. Non-compliances must be resolved according to the discrepancy resolution policy and the procedure that is to be formulated in the quality management system.

The System DDP is the pivotal document containing the definition, and all relevant references, of the system. The DDP also describes limitations with respect to the use of the system, as well as any special procedures related to the operation and use of the system. The DDP is an input to the aircraft layer flight safety documentation. The final issue of the DDP is the reference document for the air vehicle layer certificate of design.

The System Layer Verification Report summarises the requirements traceability and verification results, using the requirements database.

5.4.3.8 Subsystem verification processes

The subsystem verification process flow is shown in Figure 79. Development of the subsystem verification specification commences in parallel with the subsystem synthesis processes. When all the requirements associated with the subsystem build is at "implemented" status, the system is subjected to tests to verify that the subsystem meets with the requirements. The subsystem verification tests are normally performed on a fully integrated system (LRU), and these verification tests are conducted in parallel with the system layer tests. The test results are reviewed and collated and the verification evidence references are linked to requirements in the requirements database.

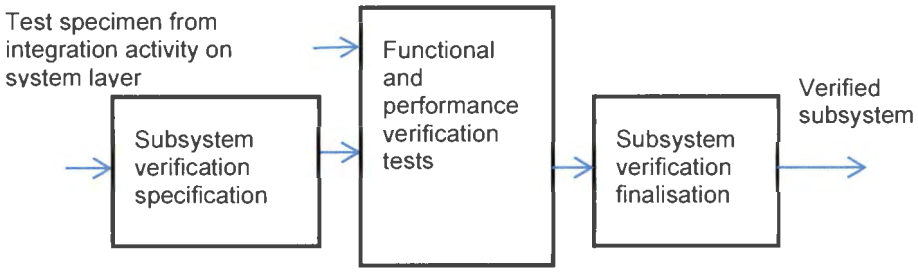

Figure 79: Subsystem layer verification process functions

The lifecycle data produced by the verification processes are listed in Table 19.

Table 19: Verification process lifecycle data

No	Item	Objective	Reference	Applicability	Notes
1	Test case description	Describes the purpose of each test case, set of inputs, conditions, expected results to achieve the required coverage criteria, and the pass/fail criteria	RTCA/DO-178C Section 11.13 b)	AV system Product Hardware Software	Same format for qualification and production; rig and air vehicle acceptance tests.
2	Test procedure	The step-by-step instructions for how each test case is to be set up and executed, how the test results are evaluated, and the test environment to be used	RTCA/DO-178C Section 11.13 c)	AV system Product Hardware	Same format for qualification and production; rig and air vehicle acceptance tests.
3	Test report	Conveys the test results of a set of test cases	RTCA/DO-178C Section 11.14	AV system Product Hardware	The test report typically contains a filled out copy of the test procedure document.
4	Flight test specification	Flight test requirement specification where MoC is flight test	RTCA/DO-178C Section 11.13 b)	AV system Product	Engineering document attached to flight test request.
5	System safety assessment	Verifies that safety requirements were satisfied	SAE ARP4761	AV system Product	May be produced in stages during development.
6	Software accomplishment summary	The primary data item for showing compliance with the Plan for Software Aspects of Certification	RTCA/DO-178C Section 11.20	Software	May be produced in stages during development.

No	Item	Objective	Reference	Applicability	Notes
7	Hardware accomplishment summary	The primary data item for showing compliance with the Plan for Hardware Aspects of Certification	RTCA/DO-254 Section 10.9	Hardware	May be produced in stages during development
8	PSS qualification report	The primary data item for showing compliance with the PSS requirements		PSS	May be produced in stages during development
9	Environmental test reports	Record environmental test reports	RTCA/DO-160	Product Hardware Software	
10	Analysis report	Record analysis results where analysis was the MoC	RTCA/DO-178C Section 11.14	AV system Product Hardware Software	Use engineering report template and instructions
11	Review report	Record review results where review was the MoC	RTCA/DO-178C Section 11.14	AV system Product Hardware Software	Use observation and action register template and instructions
12	System verification report	Summarises system level requirements compliance		AV system Product	Associated with Functional Configuration Audit results
13	Engineering report	Record results of a technical investigation or analysis	N/A	All	To be used for all types of engineering reports which are used for verification evidence or which are to be distributed to external parties

5.5 Controls

Two classes of controls have been identified in the derivation of the applied process model, i.e. configuration management and validation.

5.5.1 Configuration management and change control

The configuration management process manages lifecycle data associated with a determinable baseline. In the process definition presented here, baselines are associated with sets of requirements and the lifecycle status of the requirements, rather than with conventional (military standard) project milestones that are appropriate for large systems. The primary motivation for this approach is that the system functionality is essentially achieved by embedded software, where an increment in functionality can be developed while a verified functional baseline can be employed operationally.

For each development increment, three baselines are identified, i.e. a development baseline, a verification baseline and a product baseline. The development baseline identifies the set of requirements to be implemented within a build increment, the verification baseline ensures that the artefacts and lifecycle data associated with a build increment to be verified are under configuration control and that no unauthorised changes can be made to any item, and the product baseline identifies the lifecycle data associated with the product which is released to service.

Note: The control constraints on data which do not form part of the build definition or verification trail, e.g. data associated with rapid prototypes and ad hoc integration setups, are not as stringent and should allow for more flexibility and be less restrictive than the former.

The configuration management process authorises changes to lifecycle data and controls the method by which changes are implemented. The configuration management process documentation (problem reports, implementation proposals, change authorisations and CCB meeting minutes) provide traceability of development decisions.

5.5.1.1 Control of lifecycle data

Changes are authorised by the technical authority for the project. The technical authority conducts CCB meetings where the changes are discussed amongst stakeholders. The

meeting recommends the implementation of a change. The decision on the implementation still remains with the technical authority.

The process for editing lifecycle data is shown in Figure 80.

The planning and control processes within a particular project should indicate the requirement for a specific lifecycle data item to be produced. The responsible manager should decide on the person who must perform the task. This manager instructs the project configuration practitioner to issue the lifecycle data item – e.g. a prepared document template in the case of a new item, or the appropriate version of an item to be updated, to the designated person, after all configuration identification fields in the data item were populated and verified to be correct.

The designated person then prepares the draft version of the data item and after having properly proofread it, forwards the draft item to the project Quality Control (QC) officer.

The QC officer arranges for the inspection/review of the item. The review process is described in the following section. The records of the review serve as evidence of validation of the data item where appropriate. As part of the review process, an action register is produced, listing all observations and associated required changes to the lifecycle data item.

The designated person incorporates the updates and returns the item to the QC officer.

Once the QC officer has ascertained that all recommended changes resulting from the inspection process were correctly incorporated, he/she arranges for the document to be issued through the CM system, and signs off the action register.

5.5.1.2 Tracking requirements implementation

The CM process employs the formalised requirements to control the synthesis and verification processes. The implementation of each requirement is tracked according to the following lifecycle:

a) Draft
b) Validated
c) Implemented
d) Verified
e) Closed

In order to change the status of a requirement to "validated", and thereafter for any progression of the status, CCB approval is required. Typically, groups of requirements will be presented at

a CCB for status change approval. The CCB approval also identifies the synthesis and verification lifecycle data which are associated with the requirements lifecycle status change.

5.5.1.3 Indexing of lifecycle data

At the stage where the requirements are approved as to be validated, the Product MRI should be initiated. As lifecycle data is developed in downstream activities, the MRI is updated to keep track of all lifecycle data pertaining to the specific build of the system-of-interest. Note that the equivalent for an MRI on the hardware layer is a top-level drawing, and in the case of software it is the Software Configuration Index (SCI). For software, the configuration of the development environment is captured in a Software Lifecycle Configuration Index (SLCI).

5.5.1.4 Insular debugging process

The insular debugging process is used during development activities prior to the establishment of a verification baseline to record problems identified during e.g. integration and to keep track of the corrective actions. This process is used when different persons perform e.g. integration and software coding tasks.

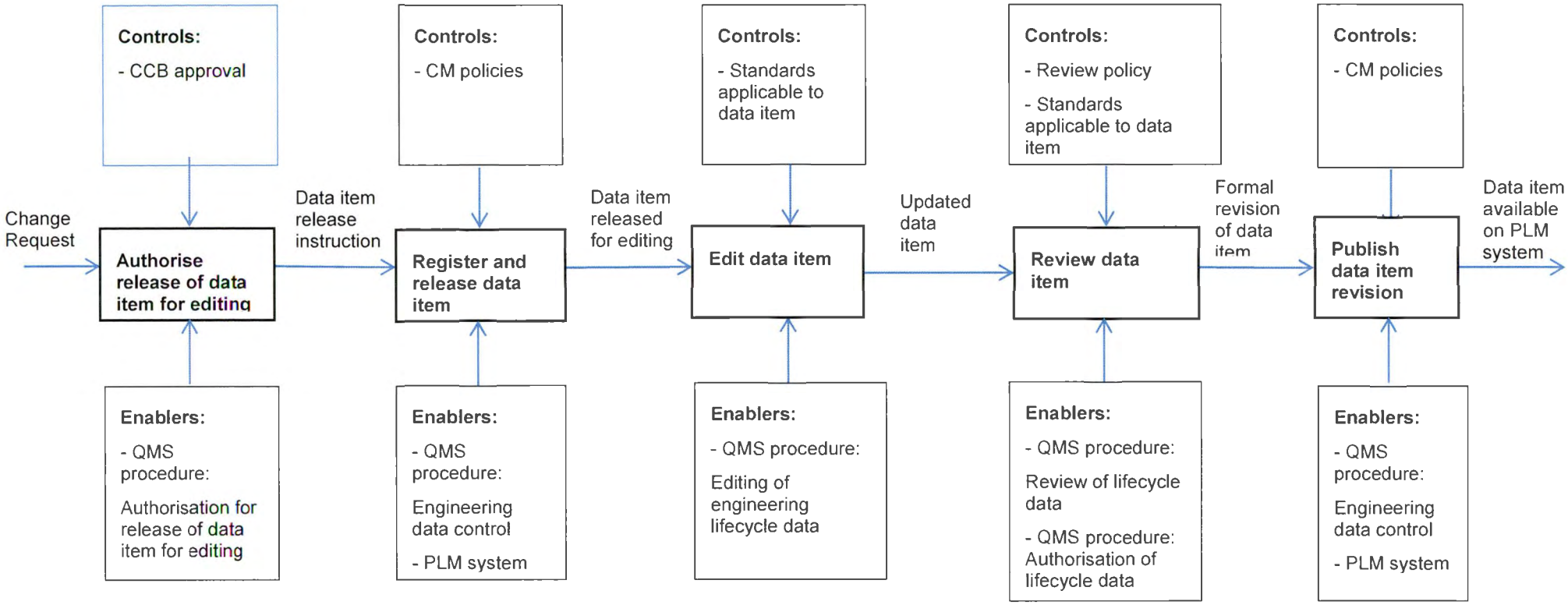

Figure 80: Process for editing of lifecycle data

5.5.2 Validation processes

5.5.2.1 Requirements validation

Requirements validation has a number of aspects that need closer examination.

The first stage of validation is the determination of the reasons for the existence of the requirement. If a requirement is directly traceable to a higher layer (e.g. avionics system or air vehicle), then the requirement can be considered to be validated. For instance, where this is not the case, a specific analysis is required to provide the rationale for the requirement. Engineering reports should be generated to record the outcomes of a traceability analysis and of the requirements analysis in the case where traceability cannot be shown. A field in the requirements database should make provision for the recording of the rationale or for cross referencing to the engineering reports.

The second stage of validation is the confirmation of the correctness of the outputs of the requirements management process. This validation normally takes the form of a formal review of the entries in the requirements database and the engineering reports. During this review, every requirement statement shall be assessed to ensure conformance to the applicable requirements standard, compliance and traceability to associated parent requirements, and that inadequate or incorrect data is corrected. Depending on the details in the agreement, this review may include the acquirer as well.

5.5.2.2 Review of lifecycle data

Product consistency is ensured by the validation of process function outputs, typically by means of lifecycle data reviews and quality assurance audits.

All lifecycle data related to the definition of a development increment shall be subjected to a structured formal review process under auspices of the designated quality control representative.

Internal standards are developed to provide guidelines by which product consistency can be assured. These include requirements and design standards on all layers of the hierarchy, and software coding standards. These standards are updated as technology evolves.

During the review, the item (document, database extract, source code, physical artefact) is inspected against applicable standards to ensure consistency of the product. The review is also used to validate the content of the item, e.g. the design of an electronic circuit is reviewed by peers, where applicable.

The review report and the associated action register form part of the quality assurance records for the project and is referenced as validation evidence where appropriate.

Approval, i.e. confirmation of validation of a lifecycle data item implies that the item does not contain errors and that it fulfilled its objectives. The review process should also meet with the requirements for the approval of the item reviewed, i.e. the document should be considered "approved" after completion of the review. This means that a reviewed item does not have to be "circulated" for approval signatures, and it further implies that all stakeholders with respect to the item must participate in the review.

If the item requires customer approval, then the internally approved item should be issued to the customer for approval at a specific revision status. Customer feedback should be managed using the change control process, where the change is initiated by a problem report citing customer input. The final document is issued at a new revision status.

Note that this process meets with all the objectives of conventional design reviews and configuration audits.

5.5.2.3 Process assurance

Process assurance audits are conducted to confirm that the identified processes were followed in producing the system. The audit reports serve as evidence of process adherence, which is required for airworthiness approval purposes.

Verification tests are witnessed by an internal (Denel Aviation) QAR as well as by a QAR nominated by the customer. The QARs sign off on the test results, indicating validation that the tests were performed according to the test procedures and that the results were recorded correctly. In the case of verification flight tests, the flight test crew fulfils the role of the QARs.

5.5.2.4 Validation of verification process functions output

The Test Reports are reviewed by the normal review process. The review results provide confidence that the test results are correct to achieve the RTCA/DO-178C requirement from Section 6.4.5.c of the standard (that test results are correct and discrepancies explained). Test reports are not subjected to change control. A test report cannot be updated. If a test is repeated, a new test report with a different configuration number is generated.

In all cases where rig test results are presented as verification evidence, the rig shall be subjected to a verification process. This implies that the rig should be treated as a system on its own and be subjected to the generic system development process comprising requirements, synthesis and verification lifecycles. The test setup shall be under configuration management.

5.6 Enablers

The enablers are the mechanisms required to facilitate the process. Enablers are categorised as either methods or tools. The enablers that are frequently encountered in the development process are described below.

5.6.1 Planning

Development plans are used to communicate project specific details to stakeholders. The plans address items (such as scope, tasks, methods, tools, risks and resources) that cannot be described in a generic process description. Plans augment the policies and procedures defined in the QMS.

The planning process includes the development and maintenance of internal standards. Note that task outputs are validated against these documents, i.e. they enable the validation process. The plans are developed at the outset of the project and are updated when required, in order to communicate project details to stakeholders.

Table 20: Planning process lifecycle data

No	Item	Objective	Reference	Applicability	Notes
1	Systems engineering plan	The SEP defines the project organisation and technical control procedures and identifies essential engineering activities and transition criteria.	INCOSE Handbook Section 5.1.2.2	AV system Product	Incorporates classical SEMP, TEMP and Systems engineering plan; Links to the project master schedule.
2	Planning documents tree	Defines the elements and relationships w.r.t project planning documents.		AV system Product	
3	Certification plan	The certification plan describes the certification basis (applicable regulations and special conditions), defines the product and installation to be certified, outlines the product development processes to be used for development assurance, and identifies the proposed means of compliance with the regulations.	SAE ARP4754	AV system	Basis for agreements between applicant and certification authority.
4	Configuration management plan	Establish the methods to be used to achieve the objectives of the system CM process throughout the system lifecycle.	RTCA/DO-178C Section 11.4	AV system Product	See also SW and HW CM plans; same pro forma may be used.
5	System safety plan	Establish the methods to be used to achieve the objectives of the system safety process throughout the system lifecycle.	SAE ARP4754, SAE ARP4761, RTCA/DO-178C, RTCA/DO-254	AV system Product	The system safety plan must address the means by which the system safety objectives described in the references will be achieved.

No	Item	Objective	Reference	Applicability	Notes
6	Quality assurance plan	Establish the methods to be used to achieve the objectives of the quality assurance process throughout the system lifecycle.	RTCA/DO-178C Section 11.5	AV system Product	See also SW and HW CM plans; same pro forma may be used.
7	Plan for hardware aspects of certification	Defines the processes, procedures, methods and standards to be used to obtain certification authority approval for certification of the system containing hardware items.	RTCA/DO-254 Section 10.1.1	Hardware	The PHAC represents an agreement between the certification applicant and the certification authority on the processes and activities to be conducted and the resultant evidence to be produced to satisfy the hardware aspects of certification.
8	Hardware design plan	Defines the processes, procedures, methods and standards to be used during the design of the system containing hardware items.	RTCA/DO-254 Section 10.1.2	Hardware	Shares significant contents with PHAC.
9	Hardware validation plan	Defines the processes, procedures, methods and standards to be used for the validation of hardware item derived requirements.	RTCA/DO-254 Section 10.1.3	Hardware	Same purpose as PSAC, applicable to electronic hardware.
10	Hardware verification plan	Defines the processes, procedures, methods and standards to be used for the verification of hardware items.	RTCA/DO-254 Section 10.1.4	Hardware	
11	Hardware configuration management plan	Describes the methods to be used to achieve the objectives of the hardware CM process throughout the system lifecycle.	RTCA/DO-254 Section 10.1.5	Hardware	
12	Hardware process assurance plan	Describes the methods to be used to achieve the objectives of the hardware QA process throughout the system lifecycle.	RTCA/DO-254 Section 10.1.6	Hardware	

No	Item	Objective	Reference	Applicability	Notes
13	Plan for software aspects of certification	The PSAC is the primary means used by the certification authority for determining whether an applicant is proposing a software lifecycle that is commensurate with the rigor required for the level of software being developed.	RTCA/DO-178C Section 11.1	Software	The PSAC represents an agreement between the certification applicant and the certification authority on the processes and activities to be conducted and the resultant evidence to be produced to satisfy the software aspects of certification.
14	Software development plan	Describes the software development procedures and software lifecycle(s) to be used to satisfy the software development process objectives.	RTCA/DO-178C Section 11.2	Software	Shares significant contents with PSAC
15	Software verification plan	Describes the verification procedures and software lifecycle(s) to be used to satisfy the software verification process objectives.	RTCA/DO-178C Section 11.3	Software	Used to define verification organisation, verification methods and environments and strategies of dealing with other verification topics
16	Software configuration management plan	Describes the methods to be used to achieve the objectives of the software CM process throughout the system lifecycle.	RTCA/DO-178C Section 11.4	Software	Used to define CM environment and CM activities.
17	Software quality assurance plan	Describes the methods to be used to achieve the objectives of the software QA process throughout the system lifecycle.	RTCA/DO-178C Section 11.5	Software	
18	Hardware requirements standard	Defines the methods, rules, and tools to be used to develop the requirements.	RTCA/DO-254 Section 10.2.1	Hardware	

No	Item	Objective	Reference	Applicability	Notes
19	Hardware design standard	Defines the rules, procedures, methods, and criteria for hardware design, validation, verification, assurance and control processes and are used to assess the acceptability and quality of hardware design results.	RTCA/DO-254 Section 10.2.2	Hardware	
20	Validation and verification standard	Defines the methods, rules, tools and criteria for validating and verifying the hardware design and implementation.	RTCA/DO-254 Section 10.2.3	Hardware	
21	Hardware archive standard	Defines the methods, rules, tools and criteria used to retain and archive product data and develop and maintain project archives.	RTCA/DO-254 Section 10.2.4	Hardware	
22	Software requirements standard	Defines the methods, rules, and tools to be used to develop the high-level requirements.	RTCA/DO-178C Section 11.6	Software	
23	Software design standard	Defines the methods, rules, and tools to be used to develop the software architecture and low-level requirements.	RTCA/DO-178C Section 11.7	Software	
24	Software coding standard	Defines the programming languages, methods, rules, and tools to be used to code the software.	RTCA/DO-178C Section 11.8	Software	Define per coding language.
25	System layer requirements standard	Defines the methods, rules, and tools to be used to develop the system requirements.	RTCA/DO-178C Section 11.6	AV system Product	Use software engineering body of knowledge to develop system layer standard.

No	Item	Objective	Reference	Applicability	Notes
26	System layer design standard	Defines the methods, rules, and tools to be used to develop the system architecture and low-level requirements.	RTCA/DO-178C Section 11.7	AV system Product	Use software engineering body of knowledge to develop system layer standard.
27	Validation and verification standard	Defines the methods, rules, tools and criteria for validating and verifying the system design and implementation.	RTCA/DO-254 Section 10.2.3	AV system Product	

5.6.2 Methods

Methods are typically described in plans, or in procedures and works instructions recorded in the QMS.

The QMS controls the set of policies and procedures applicable to the development process and ensures adherence to these policies and procedures by means of quality assurance audits.

In this process view, prototyping is considered to be an enabling method by which answers to development questions are determined.

Objectives of conventional formal prototypes, as described in classical systems engineering literature, are to elicit and validate requirements, test implementation alternatives and strategies and to demonstrate specific capabilities. The scheduling of the project according to the conventional model assumes the existence of these prototypes at the different phases of development. In cases where requirements are adequately defined and implementation technologies are obvious, or where requirements can be refined by means of e.g. simulation, no need for a series of prototypes exists.

Modelling, simulation and prototyping are hence viewed as enabling processes and their outcomes are enabling products. A significant benefit of this view is that the configuration management process is less demanding with respect to resources, and changes can be accommodated rapidly.

As a side note, it must also be understood that software generally does not get prototyped; it gets repaired until it works correctly. This notion is somewhat contrary to established principles, but observations of practical cases provide ample support for the argument.

5.6.3 Tools

A development environment provides process specific infrastructure, tools and equipment that need to be made available and maintained for use by the development project. These include requirements management tools, software development tools, test and integration rigs and test aircraft.

The requirements management system supports the requirements management process. The requirements management system consists of policies and procedures in the QMS, as well as a set of software tools.

The configuration management system supports the configuration management process. The configuration management system consists of policies and procedures in the QMS, as well as a set of software tools. The system currently employed by Denel Aviation is Enterprise Informatics eB™.

Table 21: Enabling process lifecycle data

No	Item	Objective	Reference	Applicability	Notes
1	Software lifecycle environment configuration index	Identifies the configuration of the software lifecycle environment.	RTCA/DO-178C Section 11.15	Software	This index is written to aid reproduction of the hardware and software lifecycle environment for software regeneration, re-verification, or software modification.
2	Test system description	Identifies the configuration of the system test environment.		AV system Product	Defines configurations of the environments used for verification purposes Note that the test system can also be treated as a system on its own, with associated lifecycle data.
3	QMS policy	A principle or rule to guide decisions and achieve rational outcomes.	SAE AS9100 Clause 4	AV system Product Hardware Software	A policy is a statement of intent, and is implemented as a procedure or protocol. http://en.wikipedia.org/wiki/Policy
4	QMS procedure	A document produced in support of a policy, describing the means by which a specified objective should be achieved.	SAE AS9100 Clause 5	AV system Product Hardware Software	A procedure is a series of steps to be followed as a consistent and repetitive approach to accomplish an end result. http://www.bizmanualz.com/blog/whats-the-difference-between-policies-and-procedures.html
5	QMS work instruction / manual	A document produced in support of a policy, describing the means by which a specified objective should be achieved.	SAE AS9100 Clause 5	AV system Product Hardware Software	Work instructions and manuals are used as for procedures, but compliance is not audited by accreditation agencies.

6	Engineering report	Record results of modelling, simulation or prototyping task.	N/A	All	To be used for all types of engineering reports which are used for verification evidence or which are to be distributed to external parties.

5.7 Conclusion

The detailed design for a process for the development of airborne electronic equipment in the South African context was presented in this chapter.

The context of the process within the enterprise, as well as interfaces with other processes within the enterprise, was defined.

The detailed design of the process, at the layer of the system-of-interest, was developed, and segregated into requirements, synthesis, and verification function groups. Controlling and enabling mechanisms were described.

Chapter 6 Validation of the research

6.1 Validation of the research problem

The research problem was validated from observations in real-world projects, systems engineering standards, airworthiness recommended practises, and published literature. From these sources, it was determined that no comprehensive process description for development of electronic equipment which must satisfy airworthiness requirements existed which could be implemented directly. These shortfalls were translated into research challenges, each of which was described in Chapter 3. The research problem validation is summarised in the top section of Table 22, where the research challenges are mapped to validating information sources. The research problem was further validated in the literature study in Chapter 4, where the complexity of the factors that have an influence on the design process were corroborated. This is also represented in Table 22, where arrows from literature focus areas show which sources contributed to validate research challenges.

6.2 Verification of the research solutions

Research solutions were defined to address research challenges. These solutions were determined and verified from literature in specific focus areas. Observations from literature in the focus areas were discussed in Chapter 4. Table 22 shows the research solutions that were derived from the research challenges, using the information from literature focus areas. Additional validation of the research challenges is shown in Table 22, with the arrows indicating which literature focus areas contributed to validate research solutions.

Table 22: Research verification and validation

Chapter 3	Development process evolution assessment	↓	↓	↓	↓
	AFCS development assessment: Development strategy		↓		↓
	AFCS development assessment: Software development			↓	
	AFCS development assessment: Qualification and airworthiness certification		↓	↓	
	AFCS development assessment: Change management				↓
	Oryx navigation system development assessment: Development process model			↓	
	Navigation system development assessment: HMI definition		↓		
Chapter 4	**Information sources** / **Research challenges** / **Literature focus areas**	The scope of the engineering process for the development of airborne electronic equipment is not pertinently established.	The activities and tasks required for the development of platform specific airborne electronic systems are not identified explicitly.	A lifecycle model suitable for the development of airborne electronic equipment, in particular resolving the problem of information feedback, is not determined.	The development process control mechanisms are ineffective.
	Classical systems engineering standards	↑	↑	↑ ↓	↑
	Contemporary systems engineering standards	↑	↑ ↓	↑ ↓	↑
	Quality management standards	↑	↓		↓
	Configuration management standards				↑ ↓
	Airworthiness recommended practises	↑ ↓	↓		
	Lifecycle models described in academic literature			↑ ↓	
Chapter 5	**Literature focus areas** / **Research solutions** / **Process design**	A process context is delineated for airborne electronic equipment.	Generic activities and tasks required for the development of airborne electronic equipment are identified.	An appropriate lifecycle model for the development of airborne electronic equipment is identified.	Suitable development control mechanisms are identified.
	Detailed contextualisation	↑			↑
	Development of a process framework			↑	↑
	Detailed process design		↑	↑	
	Process function details		↑	↑	
	Process controls	↑			↑
	Process enablers	↑	↑		
	Completely integrated process description	↑	↑	↑	↑

6.3 Validation of the synthesised development process

It is evident that the correct method by which the process design can be validated is by applying the process to a development project and to evaluate the results. Experience has, however, shown that the development period for typical projects for the development of airborne electronic equipment at Denel Aviation exceed three years or more. This approach to the validation of the process is impractical for the purpose of this study, and the validation of the process design had to be achieved by alternative means.

Therefore, the process for acceptance of the process description in the Denel Aviation Quality Management System (QMS) was used to validate the process description.

The QMS, described in Chapter 4.5.1 of this dissertation, contains policies, procedures, work instructions and manuals that describe the methods by which Denel Aviation conducts its business. Any of these elements is only incorporated after a formal peer review of the relevant data item has been conducted. The document that describes the process for development of airborne electronic systems, i.e. the "DSR artefact" that was developed in this study, was published in the form of an Engineering Manual [64] and incorporated in the Denel Aviation QMS. For this document to be accepted into the QMS, it had to be reviewed and approved by a panel of subject matter experts. These reviews are conducted using the same review process that applies to e.g. software inspections. The panel of experts is appointed ad-hoc, for each document (manual, policy, procedure, or work instruction) to be reviewed.

The review, or inspection process is summarised below.

- The inspection is planned by the inspection leader;
- The material for inspection is provided to the designated reviewers;
- The reviewers each study the material and record comments in individual observation and action registers;
- At a review meeting or, due to the bulk of the material as in the instance of this study,, a series of review meetings, the observations recorded by the reviewers are discussed in an orderly manner, and resolutions to each observation are recorded under control of the inspection leader;
- The author then updates the inspected item to incorporate the recommended changes, and forward the updated item to the inspection leader, who confirms that all recommended changes were incorporated satisfactorily;

- The inspection leader provides the relevant quality management practitioner with the updated document and associated inspection records, who then integrates the document in the QMS.

A listing of the reviewers who participated in the review process, as well as a summary of their credentials, is presented in Table 23.

Table 23: List of reviewers

	Functional area	Professional qualifications / registrations	Experience (Years)	Expertise and experience
Inspection leader	Systems Engineering: Quality Control	BEng (Electronics and electrical) (RAU)	> 15	Electrical 3D Modelling on weapons pylon development program. Hardware integration and software testing on Gripen fighter aircraft software development simulator. Exposed to various aspects of systems engineering and quality management processes at Saab (Sweden). Performed engineering tasks on various aspects concerning telemetry systems, data processing and analysis, During Rooivalk qualification flight test program. Responsible for systems engineering department quality control on various platforms: Rooivalk, Oryx, Cheetah.
Reviewer A	Systems engineering Program & facility management	BEng (Electronics and electrical) (RAU)	> 30	Integration and acceptance testing of fire control radar, electronic warfare and communication system on Cheetah C aircraft. Involved in concept studies for avionics systems for envisaged SAAF programs ((lead-in fighter trainer, Technical and line management for Rooivalk avionics development and qualification. Co-ordination of SA avionics delegation team activities for A109LUH helicopter development at Agusta's facility (in Italy). Executive manager for Rooivalk development.

	Functional area	Professional qualifications / registrations	Experience (Years)	Expertise and experience
Reviewer B	System safety	BEng (Mech.) (US)	> 20	SAAF System Safety Engineer (SSE) as part of development team on a number of development programs. SAAF Engineering member on Board of Inquiry into Mirage F1-AZ 235 accident (Potch 1994). Denel Aviation SSE on Alouette NOTAR (No Tail Rotor) helicopter, the Rooivalk Attack Helicopter, Mokopa Anti-Tank Missile System Air vehicle integration, Mi-24 Helicopter Weapons-, Visionics- and Avionics upgrade. Investigation leader on a number of incidents involving the operational use of the Rooivalk helicopter. Conducted and/or approved design safety assessments for all Rooivalk systems as well as upgrades on other aircraft types. Lead engineer for development flight testing of external fuel tank functionality for Rooivalk. Certification Engineer for obtaining type certification for Oryx Drummer II Navigation and Communication systems upgrade. Acting Chief Design Engineer for rotary wing unmanned aerial vehicle being developed by Denel Aviation in collaboration with various local universities.

	Functional area	Professional qualifications / registrations	Experience (Years)	Expertise and experience
Reviewer C	Systems engineering	BEng (Electronics) (PU for CHE)	> 30	Responsible for the design, build and evaluation of a four-mirror laser gyro for use in an Inertial Navigation System. Responsible for the original conceptual design of the Avionics and Weapons system of the Rooivalk attack helicopter. Systems engineer on development programs w.r.t. four separate low-earth satellites (Houwteq and SunSpace) each with different operational requirements (spy, earth resources and global navigation). Systems engineer for two different armoured troop carriers. Design, build and population of a Requirements Management System (containing about 10000 requirements) for Oryx navigation system upgrade. Investigation and resolution of several involved integration problems during development of Rooivalk and Cheetah avionics and weapons systems.
Reviewer D	Quality assurance Systems engineering Software qualification	BEng (Chemical Engineering) (UP) Diploma in Data Metrics (UNISA) Certified Lead Auditor registered with Rowe Parker Associates, UK	> 20	Process engineering tasks in mining and steel manufacturing industries. Quality Manager for a software house and developed a quality management system compliant to ISO 9001. Quality assurance during the development of the Rooivalk AH-2A. In particular, responsible for software quality assurance against RTCA DO 178B. Systems engineering tasks w.r.t avionics upgrades on the Oryx transport helicopter and the Cheetah fighter aircraft. Tasks included requirement management, system design, verification and on-going engineering support.

	Functional area	Professional qualifications / registrations	Experience (Years)	Expertise and experience
Reviewer E	RAM engineering	BSc (Aeronautical Eng.) (Wits) GDE (Wits) Certificate in Systems Engineering (UP)	> 5	Involved in the development of systems to ensure cost effective supportability of Rooivalk, Oryx and Cheetah fleets. Directed obsolesce and operational deficiency study on Rooivalk helicopter.
Reviewer F	Systems engineering Systems qualification	MSc (Physics) (UNISA) Certified SE Professional Member SA Institute of Physics	> 10	Key member of teams responsible for systems engineering and system qualification on Rooivalk and Oryx helicopters. Tasks included requirement management and verification.

	Functional area	Professional qualifications / registrations	Experience (Years)	Expertise and experience
Reviewer G	Software quality engineering	Diploma Quality Engineering (Statistical Quality Management Institute SQMI) Higher Certificate Project Management (DAMELIN) Certificate in Software Testing (Faculty Training Institute) Certificate Requirements Management (University Pretoria)	> 20	Software and Hardware Quality Assurance on the Rooivalk, Oryx and Cheetah Projects. Software Certification Audits on Rooivalk and Oryx according to RTCA DO 178B. Quality Assurance inputs on document reviews on Rooivalk, Oryx and Cheetah. Managed the environmental qualification process on the Oryx avionics upgrade program.
Reviewer G	Airworthiness and Certification	BEng (Electronics) (US) Pr.Eng	> 30	SAAF Consulting Airworthiness and Certification Engineer responsible for Rooivalk Type certification as well as Lynx Helicopter Release to Service and Certification of numerous upgrades to a variety of SAAF aircraft.

The systems development process was verified to ensure the research problem is addressed by linking all steps of the systems development process to relevant research solutions. That is, the process was derived by employing the research solutions. The development of the systems development process was presented in Chapter 5, and the verification of this process is shown in Table 22. In the matrix, steps of the development process are listed and mapped to relevant research solutions for the sake of simplicity. The table thus verifies the process as it provides traceability, starting from the research problem, proceeding to research challenges and solutions, and finally resulting in a single solution that addresses all research challenges.

Chapter 7 Conclusion

7.1 Summary

The research problem was formulated in Chapter 1. The problem is re-stated below:

Synthesise and validate a process to ensure the cost effective development of platform specific airborne electronic equipment in the South African industrial and military environment.

The DSR methodology that was applied in this study was introduced in Chapter 2. In terms of this methodology, an innovative and purposeful artefact, in the instance of this study a development process formulation, was created and systematically evaluated. The DSR process encompasses three cycles, i.e. a design cycle that is the essential element of the DSR process, where design artefacts and processes are developed. The DSR process associates with the environment by means of the relevance cycle, which describes the establishment of the research challenges from the application domain, as well as the eventual operational validation of the design artefact. The rigour cycle associates the design cycle with the knowledge base of scientific theories and engineering methods.

In Chapter 3, retrospective assessments of two projects undertaken by Denel Aviation were used to validate the research problem and to determine research challenges, as part of the DSR relevance cycle. The need for the establishment of a documented process for the development of airborne electronic equipment, which formed the DSR process artefact, was identified. Four research challenges were identified to guide the next stage of the research process:

1. *Scope of engineering activities: Rationalise the scope of the process within the environment concerned with the development of airborne electronic equipment.*
2. *Definition of activities and tasks: Determine a set of generic activities and tasks suitable for the development of platform specific airborne electronic systems.*
3. *Development process framework: Ascertain a lifecycle model appropriate for the development of airborne electronic equipment, in particular resolving the problem of information feedback.*
4. *Development process controls: Identify appropriate development process control mechanisms.*

Systems engineering and airworthiness knowledge bases were reviewed and analysed, using the research challenges as references, to ascertain past knowledge that could be applied to the design of the process, i.e. to obtain research solutions to the stated research challenges. Lifecycle models described in academic literature were also evaluated in terms of their applicability to the development of safety critical systems. This review and analysis process formed part of the DSR rigour cycle, and was described in Chapter 4. The DSR rigour process yielded the following essential outcomes, with respect to the research challenges. These are summarised below.

7.1.1 First research challenge: Scope of engineering activities

Firstly, it was shown in Chapter 4.9.1 that the development process could be separated into project processes and technical processes. The technical processes are the processes directly engaged in the definition, realisation and evaluation of the system of interest; the project processes manage the agreements and the resources that facilitate the technical processes. The technical processes are relevant to the problem investigated in this study.

The second aspect that was determined with respect to the scope of the development process is that the concept of a "system-of-systems" allows for a system-of-interest to be separated into end products and enabling products. Each end product can be segregated into interacting subsystems, where each subsystem can again be treated as a system consisting of end products and enabling products. In terms of this concept, the development of the system-of-interest can be partitioned into different hierarchical layers, where each layer is aimed at the development of a subsystem of the system-of-interest. A generically similar development process is followed at each layer. This approach significantly simplifies the scope of the definition of the technical processes. For the type of system considered in this study, the hierarchical breakdown of the system is shown in Figure 53, indicating that the system-of-interest layer of hierarchy combines the outcomes that were developed at a software, hardware, and support layers.

7.1.2 Second research challenge: Definition of activities and tasks

An important finding was that the development process, at any layer of the system hierarchy, can be distinguished into three classes of process activities, i.e. the development and management of the system requirements, the synthesis of the system-of-interest by implementing the requirements in hardware, software and

associated lifecycle data, and the verification of the synthesised system against its requirements. These form a logical basis for the organisation of process activities, as well as for the establishment of baselines through which the development process can be controlled. This control aspect is elaborated upon in Chapter 7.1.4.

It was shown that the development process elements identified in terms of the second research challenge can be represented as sets of interconnected process functions, where each process function acts on a set of inputs to produce a set of outputs, and each process function is supported by a set of enablers and subjected to a set of controls. This concept of process functions forms an important basis for the process design that was developed in Chapter 5.

- Requirements process

The requirements process establishes all the requirements that the system-of-interest needs to comply with. These include user requirements, which typically specify the functional and performance requirements that the system needs to achieve, and it states constraints imposed on the design. The constraints may arise from e.g. operational or cost considerations. In addition, other classes of requirements, such as reliability-, availability-, maintainability-, supportability-, and producibility requirements, are formulated and recorded during the requirements process. The requirements also include certification related requirements, i.e. requirements which are specified in e.g. FARs.

A specific outcome of the study is the integration of the system safety process with the development process. In terms of the requirements process, all functional requirements are subjected to a Functional Hazard Assessment (FHA), which establishes the safety objectives for the system. Note that the FHA is treated as a process function on its own.

The formalised requirements direct the synthesis and verification processes. The recording of derived requirements, which are identified during the synthesis process, also forms part of the requirements process. The derived functional requirements are also subjected to a FHA to ascertain whether they have an effect on the safety objectives to which the design must conform. A further reason for the recording of derived requirements is to ensure that verification cases and procedures are also formulated in terms of these requirements.

A development increment is considered to be complete when it is formally demonstrated that the system-of-interest complies with the applicable requirements and that discrepancies were resolved.

- Synthesis process

The synthesis process consists of the following core process functions:

- High level design;
- Low level design;
- Realisation;
- Integration.

In the detailed design of the process, additional process functions, associated with these core process functions, were defined to rationalise the process description. An example is the process functions describing flight test aspects.

The high-level design process function is referred to as architectural design or concept design in different references. In the process definition presented in Chapter 5, the term "architectural design" is used. The objective of this process function is to develop a system structure that identifies the system elements to which the system requirements are allocated.

The outputs of the low level or detailed design process function are detailed design descriptions; detailed specifications for system elements to be procured; and detailed design of software algorithms.

Realisation process functions at the different layers of the system hierarchy include hardware manufacturing/fabrication; software coding; and the procurement of parts and subsystems.

The objectives of the integration process function are to integrate the elements that constitute the system-of-interest at a particular hierarchical layer to establish a documented system definition, as well as to produce representative artefacts that comply with the system requirements, and are suitable to be subjected to verification and evaluation. Integration process function tasks include the detection and removal of errors and deficiencies. Dry runs of verification test cases and procedures are also executed during the integration process, to ensure that the system, once baselined, will not fail verification tests, and also to remove errors in the descriptions of verification cases and procedures.

The synthesis process includes the Preliminary System Safety Assessment (PSSA) as prescribed by the system safety process, which evaluates the system to determine whether the design contains sufficient redundancy, and whether the design, process, and software assurance levels are commensurate with the safety objectives as determined by the FHA.

A synthesis process is executed at all layers of the system hierarchy, but the integrated artefact(s) at each layer is provided to the system layer for final integration.

- Verification process

The objective of the verification process is to provide objective evidence that the system-of-interest, as produced by the synthesis process, complies with its requirements. In some standards, this process is referred to as the qualification process. The verification process consists of the development and validation of verification cases and procedures; and the execution of verification tasks and the validation of the verification process results. The verification tasks can take the form of reviews, analyses and tests. Importantly, these verification tasks can only be executed on test specimens which represent the product in the state that it will be deployed on the aircraft, to render the verification data valid. The verification process also includes the System Safety Assessment (SSA), which provides substantiation that the safety objectives were met.

Note that a verification process is executed at each layer of the system hierarchy. Verification results obtained at a particular layer may also be provided to other layers, to complete the verification tasks associated with that layer, e.g. system layer flight test results may be provided to the software layer, to complete the verification activities at that layer.

The outputs of the verification processes are provided to the airworthiness authorities that certify the system-of-interest for use on board the aircraft type for which the system is intended.

- Enabling processes and systems

The requirements, synthesis and verification processes are supported, or enabled, by the use of tools and methods.

The tools include computer based product lifecycle management systems, integrated software development environments, and computer aided design tools.

Policies, procedures and works instructions recorded in the Quality Management System (QMS) of the enterprise constitute a noteworthy enabling system.

Plans and planning processes can also be considered to be enablers, as they provide guidance to various aspects of the development processes for a specific project.

A significant enabling mechanism which is identified in the study is the use of prototypes to refine requirements and to test implementation strategies. Importantly, note that prototypes are not subjected to the stringent controls, e.g. a change control process, as required for the items that form part of the formal definition of the system-of-interest. Modifications to prototypes can be implemented swiftly to support decision making.

7.1.3 Third research challenge: Development process framework

In the study is was pointed out that, in practical terms, a lifecycle model representing a particular development process is a description of the organisation of the processes and process functions (as summarised in Chapter 7.1.2) in a convenient structure; and the management of transitions between these processes and process functions.

It was shown that the lifecycle model that is most appropriate to the development of airborne electronic equipment is the pre-specified, multi-step model, where either of the two representations of this model as described in the Guide to the Systems Engineering Body of Knowledge (SEBoK) (as published in BKCSE [59] and shown in Figure 47 and Figure 48 can be applied, depending on specific needs of the relevant project. In this model, the requirements for a system are finalised and baselined (see below) in what the BKCSE terms the "definition stage", before commencement of the synthesis and verification processes, which form the "development stages". In the first representation, a sub-set of the requirements is selected and developed into a system that provides the functionality specified by the selected requirements. This "partially functioning" system is then deployed operationally while a further sub-set of requirements is selected and implemented, and the deployed system is upgraded when the development stage for this increment is completed. This cycle continues until all the requirements are implemented and verified and deployed. In the second representation, the functionality is also developed in increments, but deployment only takes place after completion of the final development stage. It was shown that this method can be aligned with the "classical" pre-specified, single step lifecycle model, where project milestones are aligned such that the classical Preliminary Design Review (PDR) is used to review the completed set of system requirements, and the

classical Conceptual Design Reviews (CDRs) are aligned with reviews of qualified (i.e. verified and validated) development increments. This reduces risks to the contracting agency and improves the efficiency of the development team.

It was also shown that this lifecycle model, used in conjunction with adequate control mechanisms, allows for the handling of feedback, i.e. information which becomes evident at later stages in the development cycle, which affects data produced by "upstream" processes. The mechanisms by which this is achieved are described in the study.

7.1.4 Fourth research challenge: Development process controls

For each process function, two classes of controls can be identified, i.e. the mechanisms which ensure that tasks apposite to the process function are executed; and those mechanisms which ensure that the tasks were performed correctly.

- Configuration management

The mechanism to ensure that appropriate tasks in terms of the development of a system are executed, is provided by the Configuration Management (CM) process. This process ensures that authorised changes are applied to the appropriate data items and that, once approved, these data items are protected against unauthorised changes. The CM process manages transitions between development stages by means of baselines. In the lifecycle model adopted for the design of the process which forms the topic of this study, the following baselines are applied:

> The Requirements Baseline is established after approval of the requirements statements by the contracting agency. This baseline presents the reference against which the synthesis and verification processes are executed.
>
> Verification Baselines are established for each development increment, after completion of integration testing of that increment, to provide a representative example of the system associated with the development increment, to be subjected to verification activities, together with the data which describe its attributes.
>
> A Product Baseline (PB) for a development increment is established after completion of the verification activities for that increment, and the validation of the verification results. The PB represents the operational configuration of the system-of-interest, for a given implementation increment.

- Validation of process function outputs

It was shown that the terms validation, verification, and qualification are not applied with consistency in the sources that were consulted in this study.

As a control mechanism, the term "validation" is used in this dissertation to describe the process by which the correctness and completeness of lifecycle data and other process outputs are ensured, and it includes the validation of requirements prior to the commencement of the implementation of the requirements and the demonstration of traceability between data items. Validation also includes the demonstration of compliance to a standard. Other typical validation activities are design reviews and document inspections, where the objective of the review or inspection is to affirm the integrity of the design or document. Validation can be achieved by means of inspections, reviews, analyses and/or tests.

It was shown that, if each output of a process function was subjected to appropriate validation processes, no additional controls are required to ensure integrity of the outcomes of the development process. If the validation tasks are performed with sufficient independence, the objectives of quality, process, and design assurance as recommended by systems engineering and airworthiness references, are implicitly achieved. In other words, the efforts required to comply with process requirements from the various stakeholders can be consolidated by merging of the validation tasks for each process function. Records of the validation process are provided to aircraft layer processes, e.g. quality assurance, configuration management and airworthiness management.

Note that the term "validation" is also used to express the process by which a system is evaluated in its target environment, to prove that it fulfils its purpose. In the scope of this study, this validation activity is used to demonstrate that the user requirements were correct, and it closes a loop that is the responsibility of the contracting agency.

Note that in the process definition developed in this study, verification denotes the process of providing evidence that a requirement has been satisfied by an element of a synthesised system, which is already described in Chapter 7.1.2.

7.1.5 Process synthesis and validation

A comprehensive process design description was developed, based on the research solutions obtained through the DSR rigour cycle. The design of this process also included the incorporation of aspects emanating from practical experience in the

development of airborne electronic equipment at Denel Aviation. The process design description that resulted from the DSR design cycle was subjected to a peer review, for validation purposes. The acceptance of the process by the review panel served to complete the relevance cycle of the DSR method.

An abridged version of the process design for the system-o-interest layer of hierarchy is presented in Chapter 5.3. The comprehensive design of the process is described in an Engineering Manual, which is under configuration management in the Denel Aviation Quality Management System. The description contained in this manual presents a detailed process design at the system-of interest; hardware; software; and product support layers of the system hierarchy.

7.2 Value of the research

The need for a detailed process description for the development of platform specific airborne electronic equipment in the South African military context was identified in Chapter 1. The development of the process description was therefore a management requirement, and had to be done as part of the day to day activities of the Engineering Department at Denel Aviation as a matter of course.

The fact that the formulation of this process description is based on academically substantiated research results, provides confidence to all stakeholders that the process design is grounded on sound theoretical foundations, and that the risks associated with the selection of an appropriate development process model are minimised.

A further positive result is that the research produced a simple visualisation of the essential elements of the development process specific to the development of airborne electronic equipment, as well as of the information flow between these elements. This significantly enhances the communication concerning development process aspects between stakeholders.

This simplification also made it possible to align the outcomes of this process, based on contemporary standards and airworthiness recommended practises, with conventional "military" engineering contracting models. This alignment provides a common reference for the establishment of agreements between the military contracting agency and contracting enterprises.

7.3 Future work

7.3.1 Wider application of the generic lifecycle model

The key feature of this study is a simplified and generic lifecycle model that was used to structure a comprehensive development process description, in this instance for the development of platform specific airborne electronic equipment. The applicability of this generic lifecycle model to process descriptions for the development of other types of systems should be explored.

7.3.2 Improvements of engineering methods identified in the study

The outcome of the study is a comprehensively documented engineering process which describes the details for all the tasks associated with the development process. This provides a framework for further improvement of the process. The fact that process functions were defined to a low level of detail facilitate improvements to be made to specific methods which were identified in the process design.

7.3.3 Development of process metrics

Process metrics were explicitly excluded from the study in order to narrow the scope of the study. The low level of detail should facilitate project planning to a level of detail which can support the development of metrics for all the elements of the process, to improve the planning and control of future projects.

7.4 Conclusion

The research conducted in this dissertation was summarised in this chapter.

A summary of further work that should follow this research was presented.

Chapter 8 References

1. Lena Dahlberg and Colin McCaig: Practical Research and Evaluation – A Start-to-Finish Guide for Practitioners. Sage 2010
2. Gregor and Hevner/ Positioning and presenting Design Science Research for maximum impact: MIS Quarterly Vol. 37 No. 2—Appendices/June 2013
3. Hevner et al./Design Science in IS Research: MIS Quarterly Vol. 28 No. 1, pp. 75-105/March 2004
4. Hevner, Alan R. (2007) "A Three Cycle View of Design Science Research," *Scandinavian Journal of Information Systems:* Vol.18: Iss. 2, Article 4.
5. ARINC Specification 429, "Digital Information Transfer System (DITS)"
6. MIL-STD-1553B, MILITARY STANDARD: Aircraft Internal Time Division Command/response Multiplex Data Bus. 21 September 1978
7. Digital Avionics Systems Principles and Practise 2nd edition Gary R Spitzer Blackburn Press.
8. MIL-STD-490A, MILITARY STANDARD:: Specification Practices (04 JUN 1985)
9. MIL-STD-1521B: Technical Reviews and Audits for Systems, Equipments, and Computer Software.
10. MIL-STD-498, MILITARY STANDARD: Software development and Documentation.
11. RTCA/DO-178B: Software Considerations in Airborne Systems and Equipment Certification, December 1, 1992.
12. ANSI/EIA-632-1999: Processes for Engineering a System. (Published by Government Electronics and Information Technology Association (GEIA); January 1999).
13. Viljoen, D.A; A Process Model for the Development of Airborne Electronic Equipment. A dissertation presented to the School for Computer and Electronic Engineering North-West University November 2008.
14. DEF STAN 00-0970 Design and Airworthiness Requirements for Service Aircraft VOL 2- Rotorcraft.
15. RSA-MIL-STD-3 Issue 1. Programme Baseline Standards
16. MIL-STD-499A, MILITARY STANDARD: Engineering Management. 1 May 1974

17. MIL-STD-480, MILITARY STANDARD: Configuration Control – Engineering Changes, Deviations, and Waivers
18. MIL-STD-483, MILITARY STANDARD: Configuration Management Practises for Systems, Equipment, Munitions, and Computer Programs
19. DoD STD 2617: Defense Systems Software Development. Published 29 February 1988.
20. Kayton and Fried: Avionics Navigation Systems, Second Edition. John Wiley and Son, 1997.
21. Ritchie and Lewis / Qualitative Research Practise Sage 2003.
22. Schlager, J. (July 1956). "Systems engineering: key to modern development". IRE Transactions.
23. Joseph E. Kasser: Seven systems engineering myths and the corresponding realities. Proceedings of the Systems Engineering Test and Evaluation Conference, Adelaide, Australia, 2010.
24. RTCA/DO-160E: Environmental Conditions and. Test Procedures for Airborne Equipment, December 9, 2004.
25. RTCA/DO-178C: Software Considerations in Airborne Systems and Equipment Certification, December 13, 2011.
26. RTCA/DO-254: Design Assurance Guidance for Airborne Electronic Hardware, April 19, 2000.
27. SAE AS9100 (Revision B): Quality Management Systems – Aerospace – Requirements Revision B Revised 2004-01.
28. ISO 9000:2005 Quality management systems – Fundamentals and vocabulary
29. ISO 9001:2008 Quality management systems – Requirements
30. ISO 10007:2003 Quality management systems – Guidelines for configuration management
31. MIL-HDBK-61, MILITARY HANDBOOK: CONFIGURATION MANAGEMENT GUIDANCE (30 SEP 1997)
32. MIL-STD-973, MILITARY STANDARD: Configuration Management
33. SAE ARP 4754: Certification Considerations for Highly – Integrated or Complex Aircraft Systems.
34. SAE ARP4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.
35. Federal Aviation Administration Advisory Circular 23.1309-1E - System Safety Analysis and Assessment for Part 23 Airplanes

36. Federal Aviation Administration Advisory Circular 25.1309-1A –System Design and Analysis
37. MIL-STD-882, MILITARY STANDARD:: Department of Defence Standard Practise – System Safety: 11 May 2012
38. IEEE 1220–2005: IEEE Standard for Application and Management of the Systems Engineering Process
39. ISO/IEC 15288:2008: Systems and software engineering – System lifecycle processes.
40. ISO/IEC TR 24774: systems and software engineering – lifecycle management –Guidelines for process description.
41. ANSI/EIA-649-1998: National Consensus Standard for Configuration Management. (Published by Government Electronics and Information Technology Association (GEIA); July 1998).
42. ISO/IEC/IEEE 29148:2011: Systems and software engineering – System lifecycle processes. – Requirements engineering
43. Estefan, J A: Survey of Model-Based Systems Engineering (MBSE) Methodologies. INCOSE MBSE Initiative
44. Blanchard, Fabrycky: Systems Engineering and Analysis (Fourth Edition) Prentice Hall.
45. ADB062457: Integrated Computer-aided Manufacturing (ICAM) Architecture Part II Volume IV – Function Modelling Manual ($IDEF_0$). Air Force Wright Aeronautical Laboratory June 1981.
46. MIL-STD-881, MILITARY STANDARD: Work Breakdown Structure for Defense Materiel Items
47. IEEE/EIA 12207.0: Standard for Information Technology-Software Lifecycle Processes.
48. INCOSE Systems Engineering Handbook version 3: International Council on Systems Engineering.
49. Guideline for identification of process components CMMI-SE/SW, v1.1 Continuous Representation.
50. FAR Part 21: Certification procedures for products and parts.
51. DEF STAN 00-56 Safety Management Requirements for Defence System Part 1 Requirements
52. MIL-STD-961E Defense and Program-Unique Specifications Format and Content

53. MIL-STD-1521B (USAF) TECHNICAL REVIEWS AND AUDITS FOR SYSTEMS, EQUIPMENTS, AND COMPUTER SOFTWARE.
54. RTCA/D0-160D Environmental Conditions and Test Procedures for Airborne Equipment
55. MIL-STD-490A, MILITARY STANDARD: SPECIFICATION PRACTICES (04 JUN 1985)
56. FAR 29 (Amendment 16) Airworthiness standards: Transport category rotorcraft
57. Beck, Kent et al "Manifesto for Agile Software Development" (http://agilemanifesto.org/)
58. Larman, and Basili: Iterative and Incremental development: A Brief History. IEEE Computer, June 2003
59. BKCASE Editorial Board. 2015. The Guide to the Systems Engineering Body of Knowledge (SEBoK), v. 1.4. R.D. Adcock (EIC). Hoboken, NJ: The Trustees of the Stevens Institute of Technology. Accessed DATE. www.sebokwiki.org. BKCASE is managed and maintained by the Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Computer Society.
60. Siddiqui M S, Hussain S J, and Hussain S J: Comprehensive Software Development Model, AICCSA '06 Proceedings of the IEEE International Conference on Computer Systems and Applications
61. Bell, Thomas E., and T. A. Thayer. Software requirements: Are they really a problem? Proceedings of the 2nd international conference on Software engineering. IEEE Computer Society Press, 1976
62. Royce, Winston (1970), "Managing the Development of Large Software Systems" Proceedings of IEEE WESCON 26 (August): 1–9
63. Boehm, B, "Spiral Development: Experience, Principles, and Refinements Special Report CMU/SEI-2000-SR-008, July 2000
64. Denel Aviation, Engineering Manual: Development of Airborne Electronic Equipment, Document no B15-00-08, Revision 01. 03 March 2015.