Chapter 3: Methodology and Design

Chapter 3 details the data model design method implemented during the development of the pilot data model. This includes comparisons between the different designs and layouts which were considered for this data model. As mentioned in Chapter 2 the geodatabase design method that was implemented in this study, was a combination of the methods described by Actur and Zeiler (2004) and Buliung and Kanaroglou (2004). The 10 steps described by Actur and Zeiler (2004) were divided into the three phases described by Buliung and Kanaroglou (2004). Figure 3.1 illustrates the three design phases and their respective steps.

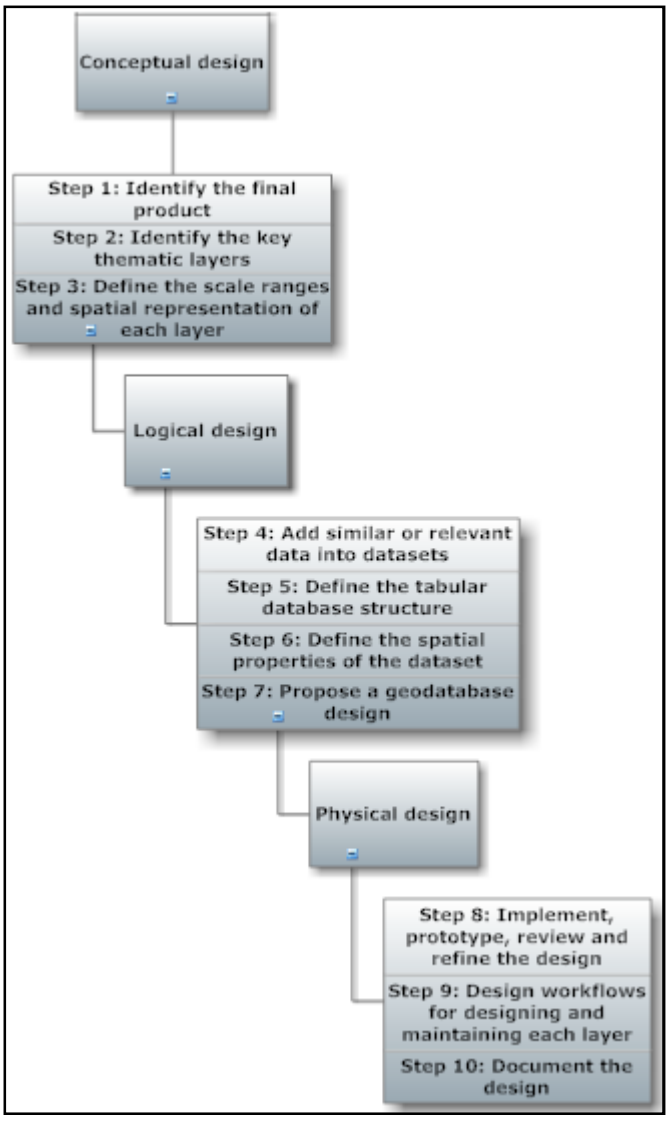

Figure 3.1: The geodatabase design phases and their respective steps

Chapter 3 describes the first 7 steps, consisting of the conceptual and logical phases, while Chapter 4 illustrates the methods which were implemented during the physical design phase. As a result, this chapter is divided into seven sections.

Section 3.1, 3.2 and 3.3 jointly describe the conceptual design step of the data model development. The aim of the conceptual design is fundamentally to create a basic understanding of what the goals of the project are; to identify which features are going to be represented; and how the features are going to be represented (geometrically) in order to achieve the project objectives.

3.1 Identify the final product

The primary aim of this study is to determine to what extent GIS software can be implemented in order to manage, analyze and visually illustrate an IT-network between buildings as well as inside of buildings on a campus. This is achieved by developing a GIS data model in order to solve IT-infrastructure management problems for a study area of three buildings. The data model should first and foremost be able to accurately represent the infrastructure of the network, outside (from the source) as well as inside buildings.

Outdoor hardware is easily represented on top of a campus satellite image in a 2D environment. The indoor hardware however is complicated due to the fact that buildings may consist of multiple floors. If for instance two features (switches) are located exactly above one another in a building, they would share the same x- and y-values, but would have different z-values (height). In some instances individual features might share a location (x-, y- and z-values). One such example is cables that share a trunking along a certain segment of a path. It would be impractical to attempt to accurately represent the cable location inside a trunking, due to the fact that multiple cables along the same route tend to be intertwined. It is of more importance from a network management perspective to be able to determine the end-points of a cable. It was decided to represent cables which share the same trunking as if they share the same space (x-, y- and z-values), this would not have an effect on the network connectivity. It is impossible to accurately represent individual features on multiple levels in a 2D environment such as ArcMap. In order to overcome this obstacle, the data

model should be able to represent not only the utility infrastructure in 3D, but the building layout as well.

A need also exist to geographically represent the connectivity of the network. Due to the lack of a modern information management tool at the Potchefstroom campus, important spatial and non-spatial information is located at various locations, and not centrally managed. The absence of such an information management application makes it difficult for the appropriate management staff to locate network elements and display its attributes as well as run analyses on the network.

The data model should be able to store and display descriptive attributes of all the elements. The data model should also contain non-spatial information tables, which relate to the spatial elements. Relationships between features; relationships between non-spatial tables; as well as relationships between a feature and a non-spatial table should be depicted in the model. Representing all of the relationships in the data model will reduce unnecessary duplication of descriptive information.

The data model also needs to store raster images. The satellite image of the campus is a raster dataset and will be used as a background which will help the user to orientate him/her self. The raster will be stored in the geodatabase. The final requirement of the data model is that all the utility features need to be spatially connected in order to create a network. Creating a network allows the user to perform analysis on the network, in order to help infrastructure management solve IT-related problems. Analyses

The next step in designing and creating a geodatabase will describe which elements will to be represented in the geodatabase, as well as which source data will be used.

3.2 Identify the key thematic layers

The following Section identifies the source data, from which the thematic layers were created. It also identifies the features which were represented in the data model. These include all the key feature classes such as buildings, rooms, utility cables and network points, providing a short description of each one.

The source data used to develop the data model consisted of two types of data. The first type of data was a referenced QuickBird satellite image of the campus, which was obtained from the university (NWU, 2011). The satellite image is stored in the geodatabase as a raster data type. The image was already correctly georeferenced and has a pixel size of 0.6m x 0.6m**.** The image serves as an orientation tool for the user, to locate the position of a specific area on campus, and does not take part in any analysis functions. The satellite image also serves as spatially referenced data, in order to correctly georeference CAD data.

The CAD data, which was utilized as source data, was obtained from the Technical Services department of the campus. Several sets of CAD data were incorporated into the development of the data model, including digital CAD data depicting the floor layout of each floor for both buildings of the study area. The building outlines; campus zones; as well as the external infrastructure were all derived from digital CAD data. The indoor network infrastructure for the ground floor of building E4 was also derived from digital CAD files, while the location for the rest of the indoor infrastructure was derived from hard copy 1:100 CAD drawings.

The campus is divided into alphabetical zones (Figure 1.1). The first thematic layer represented these zones on campus. For most part buildings from the same faculty are grouped together in a specified zone. Buildings which are not part of a faculty (such as administrative buildings) are scattered around campus. By providing the zonal information, it is easier to detect the whereabouts of a building/room/utility on campus, in terms of the campus as a whole.

The second thematic layer represented the building outlines of the campus. Creating building outlines helped to create topological rules for the utilities, which increased the data model's integrity. An example is a network port, which is not allowed to be outdoors. A

topology rule such as “network port must be covered by building”, would ensure that all the network ports would be located indoors.

A thematic layer representing rooms which are inside buildings is vital. If information only about the location of utility infrastructure inside of a building was provided, it would be a difficult task for the technician to locate the hardware. Due to the fact that a building contains numerous rooms on different levels, it becomes very important for the data model to contain a layer which represents all rooms on all floors. Each room was depicted as an individual feature in this layer, divided into floor subtypes, each with the appropriate z-values, as described by Mandloi (2007).

The data model also contained three types of utility data. Each utility type is represented in the data model as its own independent layer. The largest of these is the cables layer. Cables make out the largest part of the campus computer network and are the backbone of the system. Cables convey information between geographically distant points.

Network ports is the second utility layer, and serve as the endpoint of the network. Network ports are points where a computer can gain access to the network. Each network port is connected to a cable which conveys the information bundle to and from the accessed user. The third utility layer is switches. Switches are points in the network where the network is split in order to feed multiple endpoints or other switches. Switches are connected to one cable at the entry point, and connected to several cables at the exit point, each transmitting data to an endpoint.

The data model also contains three non-spatial data tables which provide extra information about the features. The “Owners” table contains a list of people responsible for the rooms. Information about the owners includes their name and surname, contact details and which faculty and department they work for. The second table in the data model is called the “Maintenance register”. The maintenance register is an inventory of all the maintenance jobs, installations and upgrades done on the computer network of the campus. The information contained in this register describes the date of the job; the intensity of the job and gives a short description of the job. In the same way, the data model contains a “List of

contractors" table, which contains information such as name, surname, contact details and company of the contractors which do regular maintenance work on the infrastructure.

Locating utilities inside of a building is easily done when the individual rooms are depicted; however locating infrastructure which is positioned externally is more complex. Due to a large number of external hardware placed underground. In order to overcome this obstacle, the data model needed to be able to accurately display network infrastructure outdoors. Additional data included in the data model was a fishnet raster. A fishnet is a feature class which contains a net of rectangular cells. By overlaying a grid over the satellite image, the data model accuracy increased for external hardware location, by focusing in on a small area. The final data type included in the data model is images. The data model included pictures of the switches, which were hyperlinked to the switch feature class. A picture of the switch adds additional information and establishes a visual illustration to the user/technician when searching for infrastructure (ESRI, 2011).

3.3 Define the scale ranges and spatial representation of each layer

This section assigns the names to each feature class as well as describes how the features are represented in the data model in terms of scale and shape by means of Table 3.1. In GIS the term scale can have one of two meanings. The display scale refers to the size of the map feature in terms of its real-world size at a certain zoomed extent. As the viewing extent changes (zooming in or out) the display scale also changes. Traditionally map display scales are represented by either a ration scale (e.g. 1: 50 000), a verbal scale (e.g. one centimetre represents one kilometer) or a graphic scale such as scale bars. The source scale refers to the scale of the source data that was utilized to create the layers. Table 3.1 illustrates the source scale of each layer. The table also provides a brief description of each layer and defines its data type. The conceptual design step concludes with section 3.3 (ESRI, 2011).

The logical design step describes the logical layout of the database. In this step the data are grouped into datasets, while relationship classes, topological rules and domains are established in order to enhance the integrity of the data. This step also defines attribute fields, subtypes and connected networks. The logical design step concludes by proposing a

geodatabase design. Sections 3.4; 3.5; 3.6 and 3.7 each describe a part of the logical design step.

Name	Description	Data type	Scale	Shape
PUK_Zones	Campus zones	Vector	1:100	Polygon
PUK_Buildings	Building outlines	Vector	1:100	Polygon
PUK_Rooms	Room outlines	Vector	1:100	Polygon
Cables	Network cables	Vector	1:100	Line
Network_Ports	Network ports	Vector	1:100	Point
Switches	Network switches	Vector	N/A	Point
QuickBird	Satellite image	Raster	0.6m x 0.6m	Grid
Fishnet	Rectangular grid	Vector	N/A	Line

Table 3.1: Description, data type, scale range and spatial representation of each layer

3.4 Arrange similar or relevant data into datasets

The first step to developing the logical layout of a geodatabase is to arrange similar data into the correct feature datasets, in order to be able to perform analysis on the relevant data and establish topological rules between the features. The three options that were considered when this geodatabase was created are very similar to the three geodatabase layouts presented by Mandloi (2007).

3.4.1 Option 1

The first option was to create a geodatabase for each zone of the campus. Each geodatabase contained feature datasets for each building layout, as well as a feature dataset for each building's utilities. For example if the study area has two buildings (E4 and E6), the feature datasets in the Zone-E geodatabase would be:

- Utilities_E4
- Utilities_E6
- Building_layout_E4

- Building_layout_E6

Each building layout dataset contained feature classes depicting rooms for each floor in the building. Thus if building E6 has five floors, the feature classes contained in Building_layout_E6 would be:

- E6_Floor_Ground
- E6_Floor1
- E6_Floor2
- E6_Floor3
- E6_Floor4

The utilities dataset contained feature classes for each utility type on each floor of a building. In the example of building E6, the feature classes for Utilities_E6 would be:

- Cables_E6_floor_G (Ground floor)
- Cables_E6_floor_1
- Cables_E6_floor_2
- Cables_E6_floor_3
- Cables_E6_floor_4
- Switches_E6_floor_G
- Switches_E6_floor_1
- Switches_E6_floor_2
- Switches_E6_floor_3
- Switches_E6_floor_4
- NwP_E6_floor_G (Network port)
- NwP_E6_floor_1
- NwP_E6_floor_2
- NwP_E6_floor_3
- NwP_E6_floor_4

This is a pilot study and only creates a data model for a part of the campus. However when developing this data model it was vital to keep in mind that the aim of the study is to create a data model which can be implemented and extended for the whole campus. Considering the

number of buildings and the total amount of floors for each building on campus, implementing Option 1 would create a colossal geodatabase with a vast amount of feature classes.

For example: Consider a campus has 100 buildings each with a minimum of 2 floors for each building. If Option 1 was implemented it would mean that the minimum number of rooms feature classes would be 200 (a rooms feature class for each floor of each building). This means that there would be 200 feature classes used only to represent the building layouts. The minimum number of utility feature classes for the same 100 buildings would be 600 (200 each for switches, cables and network ports) (Table 3.2). Considering only the room feature classes and the utility feature classes, this gives the theoretical geodatabase approximately 800 feature classes. This large number of feature classes could be very difficult to edit and it would also create an organizational vulnerability. Another dilemma with Option 1 is that in order to create a network or a set of topological rules, each participating feature class needs to be contained by the same feature dataset (ESRI, 2011).

Option	Number of utility feature classes	Number of rooms feature classes	Total feature classes
Option 1	600	200	800
Option 2	3	2	5
Option 3	3	1	4

Table 3.2: Number of utility and room feature classes for 100 buildings with 2 floors each

3.4.2 Option 2

In order to overcome the obstacle of not being able to create a network or a set of topological rules, which was encountered in Option 1; Option 2 requires that the building layout feature classes as well as all the utility feature classes be stored in a singular feature dataset. By creating the PUK_IT feature dataset, it becomes possible to construct a network among all the utility features classes, as well as create topological rules between all the feature classes present in the same dataset. This also presents the opportunity to form relationship classes among the features. A relationship class provides additional information about another related feature.

Option 2 also reduced the number of rooms feature classes by representing all the rooms which are located on the same floor for the whole campus, in a single feature class. For example: Rooms_floor_01 displays all the rooms in all the buildings on campus which are on floor 1. The maximum number of room feature classes for Option 2 is equal to the number of floors of the building with the most floors on campus (for this study the highest building on campus has 7 floors, but in order to incorporate possible future developments it has been decided to use 10 floors).

Consider the example described in Option 1: a campus with 100 buildings. Assume that the highest building on campus has 2 floors. This means that if Option 2 is applied to represent the theoretical campus, the maximum number of rooms feature classes would be 2 (Table 3.2). The number of rooms feature classes increases according to the number of floors of the highest building.

Option 2 also decreases the number of utility feature classes by creating a single feature class for each utility type. This modification leads to a more organized geodatabase which is easier to edit. When compared to Option 1 the total number of utility feature classes for a campus with 200 individual floors is also reduced from 600 to 3. The total number of room (10) and utility (3) feature classes is reduced from 800 for Option 1 to 13 for Option 2 (Table 3.2).

Although Option 2 is a great improvement from Option 1, it is not without flaws. Some features will be highly populated (such as the ground floor and floor 1), while other floors such as floor 7 will be sparsely populated (not all buildings have a seventh floor). Another minor flaw of Option 2 is that when a user wants to examine a floor of a certain building, the data model will depict all the rooms for the whole campus which are on that floor. Due to the fact that there are many room feature classes (2); the relationship between building and room is represented as 2 relationship classes (one for each room feature class). If the number of rooms feature classes increases, so does the number of relationship classes. In the same way, every other spatial (feature class) or non-spatial table that has a relationship with rooms will have to be represented by 2 relationship classes. Features which have relationship with rooms are: PUK_Buildings; Switches; Network ports; Owners_Table and the Maintenance_register. The conclusion is that it will take about a large number of

relationship classes in order to represent the rooms' relationships. Additional relationship classes needs to be developed in order to display the remaining features' relationships.

3.4.3 Option 3

Option 3 inherits the idea of creating a single feature dataset which contains all the feature classes from Option 2. As mentioned before, a singular dataset promotes the construction of a network and topological rules. Option 3 also incorporates the example of developing a single feature class for each utility type as in Option 2. Option 3 differs from its predecessors by creating a single feature class which depicts every room, on every floor for every building. This is achieved by employing subtypes in the feature class. A field, 'Floor**'** is added to the attributes of the rooms feature class, which describes the floor number of the room and also acts as the subtype indicator. The number of subtypes is equal to the number of floors of the building with the most floors on campus. The user is able to display the rooms according to floor subtypes if required. This alteration to Option 2 allows the number of relationship classes to be decreased to one relationship class per related feature. If Option 3 is implied for the same 200 floors example as the first two options it would have one feature class representing the rooms and another three depicting the three utility types. When considering only the room and utility feature classes, Option 3 has a total of 4 feature classes (Table 3.2).

Due to the fact that Option 3 promotes the development of a network and topological rules by storing all feature classes in the same feature dataset as well as the fact that this option minimizes data redundancy by reducing the number of feature classes and relationship classes; the decision was made to employ Option 3 as the geodatabase layout for this study. The summarized layout of the data model geodatabase is described in Table 3.3.

The data model geodatabase also employs relationship classes, topological rules, domains and a network, which are described in more detail in the following two sections. Section 3.5 defines the tabular database structure by illustrating the attribute fields of each feature class and table; specifying the ranges for the domains; describing the subtypes; and modelling the relationship classes.

Geodatabase	PUK_Geodatabase
Feature dataset	PUK_IT
Feature classes	PUK_Zones
	PUK_Buildings
	PUK_Rooms
	Cables
	Network_Port
	Switches
Tables	Owner_Table Maintenance_register List_of_Contractors

Table 3.3: Geodatabase layout

3.5 Define the tabular database structure

This section describes the structural table layouts of the geodatabase in greater detail. The section begins by describing the fields of all the feature classes. The section then also explains the non-spatial tables, specifies the domains which are implemented as well as defines the subtypes which are created for certain feature classes. The section concludes by depicting the relationships which exist among the various features in the geodatabase.

3.5.1 Feature class attributes

Each feature class in the data model contains a list attributes which provide extra information about the feature class. All of the features which are located in the same feature class, share the same list of attributes, but may have different values. There are three attributes which are present in all feature classes, namely Object_ID, Shape and Unique_ID. The Object_ID field is a primary key generator and is automatically generated by the software every time that a new feature is created in the feature class. Due to the fact that Object_ID is a primary key (each feature in the class has a unique Object_ID), it is used as the relationship primary and foreign key in most relationship classes. The Shape field is also an automatically generated field. The Shape attribute stores the geometric shape of a feature including the x-, y-, and z-values.

The Unique_ID is a type of primary key and is created for all feature classes. No two features in the whole data model geodatabase have the same Unique_ID. The concept for creating a Unique_ID was derived from the case study which was presented by Glos (2008). The Unique_ID was created in order to have an ID which describes the position and connection of a feature in a single code.

The code is hierarchical and the feature classes are divided as such. The Unique_ID depicts the route from the top to the bottom of the hierarchy. Feature classes at the top of the hierarchy serves as the source and has a short Unique_ID, while feature classes at the bottom of the pyramid have a longer route and thus a longer Unique_ID.

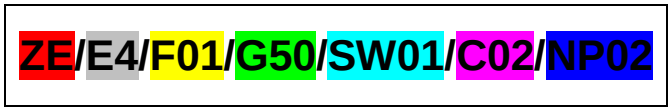

Figure 3.2: Unique_ID example

In Figure 3.2 an example of the Unique_ID is displayed. The feature class at the top of the hierarchy is PUK_Zones. PUK_Zones' Unique_ID is only depicted by the red part of the code. The second feature class in the pyramid is PUK_Buildings; its Unique_ID is represented by the red and grey parts of Figure 3.2. In the same way each following feature class in the hierarchy inherits the Unique_ID from its predecessor, while adding its own extension. Consequently the PUK_Rooms; Switches; Cables and Network_Port feature classes' Unique_ID is represented from left up to the green; light blue; pink and dark blue parts of the Unique_ID in Figure 3.2 respectively.

The Unique_ID was also created to help the user when running a query. For example if a user possesses the Unique_ID of a certain feature, and needs to locate its position, as simple Select by Attribute query will find the position of feature.

The PUK_Zones feature class depicts the alphabetical zones of the campus. This feature class' major function is to group all of the buildings on campus into the correct zones. This feature class has only two fields (except for the automatically generated fields): Unique_ID

and Number_of_Buildings (the number of buildings in the zone). The Number_of_Buildings field is incorporated into the feature class in order to provide the user with additional information about the zone. Figure 3.3 provides a summarized table of the PUK_Zones fields and their information. The *Field name* defines the name of the field. The *Data type* column specifies the type of data which the field may contain, while the *Allow nulls* column specifies if the attribute may have a null value. The *Default value* column gives the default value of the attribute. The *Domain* column specifies whether or not the attribute employs domains (and which ones).

Simple feature class
PUK_Zones

Geometry Polygon
Contains M values Yes
Contains Z values Yes

Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length
OBJECTID	Object ID						
Shape	Geometry	Yes					
Unique_ID	String	Yes					50
Number_of_buildings	Long integer	Yes			0		
Shape_Length	Double	Yes			0	0	
Shape_Area	Double	Yes			0	0	

Figure 3.3: Fields description for the PUK_Zones feature class

Simple feature class
PUK_Buildings

Geometry Polygon
Contains M values No
Contains Z values Yes

Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
Unique_ID	String	Yes					50
Zone_ID	Longinteger	Yes			0		
Name_of_Building	String	Yes					50
Faculty	Longinteger	Yes		Faculty	0		
Department	Longinteger	Yes		Department	0		
Number_of_Floors	Longinteger	Yes			0		
SHAPE_Length	Double	Yes			0	0	
SHAPE_Area	Double	Yes			0	0	

Figure 3.4: Field description for the PUK_Buildings feature class

The PUK_Buildings feature class depicts the outline of each building. As previously mentioned the building feature class contains an Object_ID and a Unique_ID field. The feature class also has a Zone_ID field, which serves as the foreign key between the PUK_Zones and PUK_Buildings feature classes (Figure 3.4). The Faculty and Department fields define which academic faculty and department the building belongs to, while the Name_of_Buildings field defines the local campus name of the building. The Number_of_Floors field is an integer value which represents the number of levels in the building. This field is utilized in order to extrude the building layer according to its number of floors, in a 3D environment.

The PUK_Rooms feature class depicts every room in the study area and is summarized in Figure 3.5. In an effort to keep this feature class organized, subtypes were implemented. The feature class contains the usual generic fields, as well as a Building_ID field. The Building_ID field is a foreign key from PUK_Buildings. The Floor field in PUK_Rooms describes the level on which the room is located and also serves as the subtype field. The Space_type attribute gives more information as to what the room function is (such as class room, office or a bathroom), while the Capacity field defines the maximum number of occupants allowed in a room. The Owner_ID is a foreign key which corresponds to the Owners_Table, which contains the information about the person responsible for the room.

Simple feature class
PUK_Rooms

Geometry Polygon
Contains M values No
Contains Z values Yes

Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
Unique_ID	String	Yes					50
Building_ID	Long integer	Yes			0		
Space_Type	Long integer	Yes		Space_Type	0		
Owner_ID	Long integer	Yes			0		
Capacity	Long integer	Yes			0		
Floor	Long integer	Yes	3		0		
SHAPE_Length	Double	Yes			0	0	
SHAPE_Area	Double	Yes			0	0	

Figure 3.5: Field description for PUK_Rooms feature class

The information technology utilities are represented by three feature classes. The Switches in the network function as an amplifier and divider of the network. The network enters a switch (via a cable) and is divided and branches out of the switch. A switch may also serve as the main source of the network. The Switch feature class contains Object_ID and Unique_ID field (Figure 3.6). The Switch also contains a Model_name field which makes use of a predefined domain to define the model of the switch. The Room_ID field serves as a foreign key for the relationship between PUK_Rooms and Switches and corresponds with PUK_Rooms' Object_ID. The maximum number of cables allowed to be attached to a certain switch is depicted by the Max_attach field. The Cable_ID field is a foreign key, which creates a relationship class between the Switch feature class and the Cables feature class. Finally the Job_ID field also serves as a foreign key to the Maintenance_register table. The Maintenance_register table is a table which contains a list and provides information of all the maintenance, upgrades and installations done on campus. The Type field defines a switch as either being the source switch, a zonal switch (star topology switch) or a buildings local

switch and serves as the subtype of the feature class. Finally, the IP_address field depicts the IP address of the switch.

Simple feature class
Switches

Geometry Point
Contains M values No
Contains Z values Yes

Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
Unique_ID	String	Yes					50
Model_name	Long integer	Yes		Model	0		
Room_ID	Long integer	Yes			0		
Job_ID	Long integer	Yes			0		
Max_attachements	Short integer	Yes	24		0		
Cable_ID	Long integer	Yes			0		
Type	Long integer	Yes	0		0		
IP_address	String	Yes					50
Source_data	String	Yes					50

Figure 3.6: Field description for Switches feature class

The Cables feature class represents the bulk of the network. The cables cover the largest geographical ground of all the utility types. The cables in the network serve as the conveyer of data bundles between network components. The Cables feature class also contains both the Object_ID and the Unique_ID fields, as seen in Figure 3.7. The Type field is the feature class' subtype and describes the material of which a cable consists, while the Standard_name field implements a domain in order to provide the standard name of a cable type. In the same fashion the IEEE_name field utilizes the IEEE_name domain in order to provide the Institute of Electrical and Electronics Engineers (IEEE, 2011) standardized name of the cable. The Max_attach, Max_length, and Max_Speed fields describe the maximum attachments to the cable; the maximum length (meters) of the cable before the network has to be divided; and the maximum data distribution (Mbps) of the cable, respectively. The Job_ID field is a foreign key which corresponds to the Object_ID field of the Maintenance_register table. The NP_ID field is a foreign key which relates to the connected Network_Port feature class. Finally the SW_ID field is a foreign key which relates to the corresponding feature in the Switches feature class.

Simple feature class
Cables

Geometry Polyline
Contains M values No
Contains Z values Yes

Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
Unique_ID	String	Yes					50
Type	Long integer	Yes	0		0		
Standard_name	Long integer	Yes		Standard_name	0		
IEEE_name	Long integer	Yes		IEEE_name	0		
Max_length	Short integer	Yes			0		
Max_attach	Short integer	Yes			0		
Max_speed	Short integer	Yes			0		
Job_ID	Long integer	Yes			0		
NP_ID	Long integer	Yes			0		
SW_ID	Long integer	Yes			0		
SHAPE_Length	Double	Yes			0	0	
Source_data	String	Yes					100

Figure 3.7: Field description of Cables feature class

The network ports serve as the endpoints of the network. It is point of access between a stand-alone component and the network. Network ports are represented by the Network_Port feature class. The Network_Port contains an Object_ID field and a Unique_ID field (Figure 3.8). The Network_Port also contains a Room_ID field which is the foreign key that relates to the PUK_Rooms' Object_ID. The Job_ID is also a foreign key to the Maintenance_register table.

Simple feature class
Network_Port

Geometry Point
Contains M values No
Contains Z values Yes

Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length
OBJECTID	Object ID						
SHAPE	Geometry	Yes					
Unique_ID	String	Yes					50
Room_ID	Long integer	Yes			0		
Job_ID	Long integer	Yes			0		
Source_data	String	Yes					100

Figure 3.8: Field description for Network_Port feature class

3.5.2 Non-spatial tables

The data model contains three non-spatial information tables. Non-spatial tables do not have geometry, and for that reason do not require shape fields. The first is the Owners-table

which contains a list of people who are responsible for rooms in the study area buildings. This table was created in order to provide extra information about the room occupants to the user. For instance if IT management needs to do maintenance on the utilities of a certain room; the Owners_Table provides the contact information of the room's "owner". The GIS software automatically creates an Object_ID field for each non-spatial table in the same fashion as it does for a feature class. The table also has an Owner_ID field, which is the owner's unique personnel number. The Owner_ID field also serves as the primary key for the relationship class created between Owners_Table and PUK_Rooms. The Owner_name, Tel_number and Email fields provide the additional information such as the occupant's name; the occupant's contact number and his/her e-mail address, respectively. Finally the Department field utilizes the Department domain in order to specify the department which employs the occupant. Figure 3.9 summarizes the Owners_Table in the same way that the feature classes were described.

Table
Owner_Table

Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length
OBJECTID	Object ID						
Owner_ID	Long integer	Yes			0		
Owner_Name	String	Yes					50
Tel_Number	String	Yes					50
E_mail	String	Yes					50
Department	Long integer	Yes		Department	0		

Figure 3.9: Field description for Owners_table

ɔne on the

IT-infrastructure of the campus. The Maintenance_register contains an Object_ID field, as well as a Contractor_ID field. The Contractor_ID field serves as a foreign key, which links every job in the table to a certain contractor in the List_of_Contractors table. The Date_of_job field represents the date that the task has been completed. Finally the Maintenance_register has an Utility_component field, which describes the component (for example a cable, switch or network port) that serves as the main focus of the maintenance job. Figure 3.10 provides a summary of the fields contained in the Maintenance_register.

Table
Maintenance_register

Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length
OBJECTID	Object ID						
Contractor_ID	Long integer	Yes			0		
Utility_component	String	Yes					50
Date_of_job	Date	Yes			0	0	8

Figure 3.10: Field description for Maintenance_register table

The final non-spatial table in the data model is the List_of_Contractors table (Figure 3.11). This table lists the technicians and contractors, which is employed by the University of the North-West to complete maintenance tasks of the campus infrastructure. This table provides information about a technician's contact information and his/her occupation. The List_of_Contractors table, in the same fashion as all the previous tables, automatically generates an Object_ID field, which serves as the primary key of the table and is utilized in the relationship class. The table contains additional information about a technician's name (Contractor_name field); his/her occupation (Occupation field); and the company which employs the technician (Company field). The List_of_Contractors table also provides the contact information of the technician. The fields: Contractor_tel_nr, Email and Fax_number represent the contractor's telephone number; his/her e-mail address as well as his/her fax number, respectively. Finally the Campus field depicts the campus where the technician normally works. The University has three campuses, and often a technician will work on more than one.

Table
List_of_Contractors

Field name	Data type	Allow nulls	Default value	Domain	Prec-ision	Scale	Length
OBJECTID	Object ID						
Contractor_Name	String	Yes					50
Occupation	String	Yes					50
Contractor_Tel_nr	String	Yes					50
Email	String	Yes					50
Company	String	Yes					50
Campus	String	Yes					50
Fax_nr	String	Yes					50

Figure 3.11: Field description of List_of_Contractors table

3.5.3 Domains

A domain describes the valid values of a certain field. A list is created for the geodatabase and may be utilized by any field which has the same data type than the employed domain. This geodatabase data model makes use of five domains. This section describes each domain in terms of the type of domain (coded or range) as well as lists the possible values. The first domain is called Faculty and describes all the possible Faculties which are present on the campus. Table 3.4 lists the possible values which are provided by the Faculty domain.

Faculty Domain	
Type	Coded
Code	**Description**
0	Arts
1	Natural Sciences
2	Theology
3	Education Sciences
4	Economic and Management Sciences
5	Law
6	Engineering
7	Health Sciences

Table 3.4: Faculty domain

Each academic faculty on campus is divided into individual departments. The Department domain is employed in order to provide the valid possible input for a field which describes a department on campus. This is the largest domain in the data model and contains 33 possible values. These include all the academic departments as well as an administrative department. Table 3.5 provides a list of the possible inputs provided by the Department domain.

Department Domain			
Type	Coded		
Code	**Description**	**Code**	**Description**
0	School of Languages	16	Potchefstroom business school
1	School of Social and Government studies	17	School of Accounting Sciences
2	School of Music	18	School of Economics
3	School of Communication studies	19	School of Business Management
4	School of Philosophy	20	Human resources sciences
5	School of Physical and Chemical Sciences	21	Centre for community law and development
6	School of Environmental Sciences and Development	22	School of Chemical Engineering
7	School of Computer, Statistical and Mathematical Sciences	23	School of Electrical, Electronic and Computer Engineering
8	Centre for Business Mathematics and Informatics	24	School of Mechanical Engineering
9	Centre for Environmental Management	25	Post-graduate School of Nuclear Science and Engineering
10	Centre for Human Metabonomics	26	School of Biokinetics, Recreation and Sport sciences
11	School of Biblical Studies and bible languages	27	School of Pharmacy
12	School of Ecclesiastical Studies	28	School of Physiology, Nutrition and Consumer sciences
13	School of Education	29	School of Psychological behavioural sciences
14	School of Continuing teacher education	30	School of Nursing
15	School of Curriculum-based studies	31	Research focus area: Teaching-Learning Organizations
		32	Administration

Table 3.5: Department domain

The Space_type domain was created with the intention to create a list of values which describe the function of each room in the PUK_Rooms feature class. When a new PUK_Rooms feature is created, the room's purpose is determined by the Space_type field,

which employs the Space_type domain. Descriptions of all the Space_type values are provided in Table 3.6.

Space_type Domain			
Type	Coded		
Code	**Description**	**Code**	**Description**
0	Office	9	Utility room
1	PC Labs	10	Empty
2	Laboratory	11	Lobby
3	Seminar/Class room	12	External area
4	Corridor	13	Elevator
5	Steps	14	Library
6	Kitchen	15	Staff room
7	Bathroom	16	Exhibition
8	Storage	17	Museum

Table 3.6: Space_type domain

Standard_name Domain	
Type	Coded
Code	**Description**
0	1000BASE SX
1	1000BASE LX
2	100BASE T
3	10BASE 5
4	10BASE 2

Table 3.7: Standard_name domain

The Standard_name domain provides a list of possible values for the Standard_name field in the Cables feature class. Every cable type has its own international standard name. This domain provides the list of appropriate standard names of the cables which are employed in this data model (Table 3.7). It also ensures that an incorrect value cannot be entered as a standard name. There are 5 different types of cables, and thus 5 different standard names in the domain. By providing a list of the correct names, the probability of the incorrect standard name being selected is decreased. Although the campus only employs three of the five

cable types; the 10BASE 5 and 10BASE 2 cables are included as subtypes to make provisions for possible future implementation.

Each cable also has a standardized IEEE name. Similarly to the Standard_name domain, an IEEE_name domain was created which contained the correct IEEE names of the cables in attendance. Table 3.8 provides a description of the individual IEEE names contained in the IEEE_name domain. Two of the cables share the same IEEE name, thus there are only 4 different IEEE names listed in the domain.

IEEE_name Domain	
Type	Coded
Code	**Description**
0	IEEE 802.3z
1	IEEE 802.3u
2	IEEE 802.3
3	IEEE 802.3a

Table 3.8: IEEE_name domain

Model Domain	
Type	Coded
Code	**Description**
0	DES 35 26
1	DES 35 28

Table 3.9: Model domain

The final domain in the data model, lists the types of switches and each has a model name. The Model domain lists the different types of switch models in order to guard the integrity of the data model. The domain ensures that only the provided list (Table 3.9) may be entered as the model name. Information describing the different types of models utilized by the University was not available. Table 3.9 however does represent the two known switch models which the University utilizes.

3.5.4 Subtypes

This data model contains several subtypes. A spatial (feature class) or non-spatial table may contain one field which subdivides the table into types, namely subtypes. Each subtype is allowed to have its own default values for the rest of the fields. Implementing subtypes into a table increases data reliability, as well as saves time with the editing process, in particular,

populating the tables. The first subtype was implemented for PUK_Rooms, and has already been explained earlier in this chapter. The subtype field for PUK_Rooms is Floor. This field divides the campus rooms into different floors. The subtypes for the Floor field are listed according to code and description in Figure 3.12.

Subtypes of PUK_Rooms

Subtype field Floor
Default subtype 3

List of defined default values and domains for subtypes in this class

Subtype Code	Subtype Description	Field name	Default value	Domain
0	Sub_Floor_01	Space_Type		Space_Type
1	Sub_Floor_02	Space_Type		Space_Type
2	Ground_Floor	Space_Type		Space_Type
3	Floor_01	Space_Type		Space_Type
4	Floor_02	Space_Type		Space_Type
5	Floor_03	Space_Type		Space_Type
6	Floor_04	Space_Type		Space_Type
7	Floor_05	Space_Type		Space_Type
8	Floor_06	Space_Type		Space_Type
9	Floor_07	Space_Type		Space_Type
10	Floor_08	Space_Type		Space_Type
11	Floor_09	Space_Type		Space_Type
12	Floor_10	Space_Type		Space_Type

Figure 3.12: PUK_Rooms subtypes: Floors

Subtypes of Cables

Subtype field Type
Default subtype 0

List of defined default values and domains for subtypes in this class

Subtype Code	Subtype Description	Field name	Default value	Domain
0	Multi-mode optical fibre	Standard_name	0	Standard_name
		IEEE_name	0	IEEE_name
1	Single-mode optical fibre	Standard_name	1	Standard_name
		IEEE_name	0	IEEE_name
2	Category 5e UTP	Standard_name	2	Standard_name
		IEEE_name	1	IEEE_name
3	Thick coaxial cable	Standard_name	3	Standard_name
		IEEE_name	2	IEEE_name
4	Thin coaxial cable	Standard_name	4	Standard_name
		IEEE_name	3	IEEE_name

Figure 3.13: Cables subtypes: Type

The second feature class to contain a subtype is the Cables feature class. The Cables feature class employs the Type field as its subtype divider. The Type field sub-divides the cables present in the feature class into each one's cable type. The Type field contains a list of codes which corresponds to a list of subtype descriptions, as presented in Figure 3.13.

Each field type has its own value for the remaining fields in the feature class, and the default value of each subtype was altered accordingly.

The final feature class to implement subtypes is the Switches feature class. This feature class utilizes the Type field as its subtype field. The Type attribute defines the function of the switch in the topological structure. It defines a switch as a source switch, a zonal switch or a local switch. The Switches feature class' subtypes are listed in Figure 3.14.

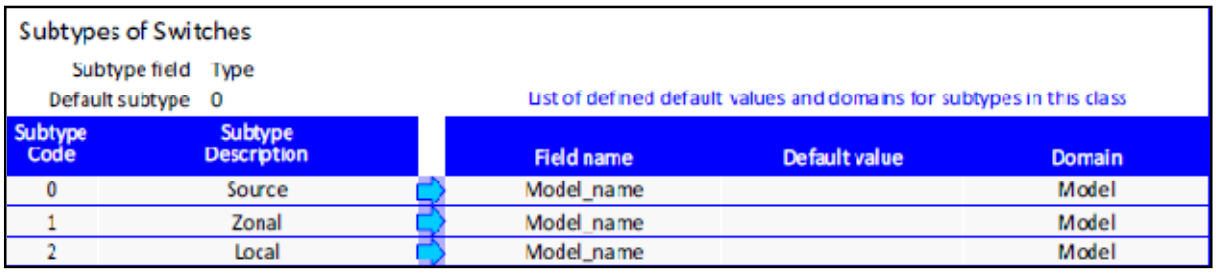

Figure 3.14: Switches subtypes: Type

3.5.5 Relationship classes

Relationship classes are important in order to reduce data repetition. A relationship class represents a link between two tables (spatial or non-spatial). The GIS software offers the user the opportunity to access a feature's information, as well access the related feature's information, without the repetition of data in the attribute tables. For example: If a network port and a cable are linked and both contain the same zonal, building or room information (geographically the same) this information would be redundant. By incorporating relationship classes only one of the features needs to contain the information which can be accessed by all its linked features, therefore reducing unnecessary information. The data model for this study has a number of relationship classes between feature classes, between non-spatial tables, as well as between feature classes and tables. Table 3.10 lists the relationship classes, and their respective tables, which are present in this geodatabase.

Relationship classes					
Name	**Origin table**	**Primary key**	**Destination table**	**Foreign key**	**Cardinality**
Zones_has_Buildings	PUK_Zones	Object_ID	PUK_Buildings	Zone_ID	1 : M
Buildings_has_Rooms	PUK_Buildings	Object_ID	PUK_Rooms	Building_ID	1 : M
Room_has_Switches	PUK_Rooms	Object_ID	Switches	Room_ID	1 : M
Room_has_NP	PUK_Rooms	Object_ID	Network_Port	Room_ID	1 : M
Owner_has_Rooms	Owner_table	Owner_ID	PUK_Rooms	Owner_ID	1 : M
Cables_has_Switches	Cables	Object_ID	Switches	Object_ID	M : N
NP_has_Cables	Network_Port	Object_ID	Cables	NP_ID	1 : 1
Switches_has_Maint	Maintenance_register	Object_ID	Switches	Job_ID	1 : M
NP_has_Maint	Maintenance_register	Object_ID	Network_Port	Job_ID	1 : M
Cables_has_Maint	Maintenance_register	Object_ID	Cables	Job_ID	1 : M
Contractor_has_Maint	List_of_Contractors	Object_ID	Maintenance_register	Contractor_ID	1 : M

Table 3.10: List of relationship classes in the data model

The data model has a total of 11 relationship classes, all of which are simple relationships. The Name column describes the name of the relationship class. The Origin table column describes the table which contains the primary key, while the Primary key column defines the field which serves as the primary key. The Destination table column in contrast represents the table which contains the foreign key, while the foreign key column describes the field which serves as the foreign key. Finally the cardinality column describes the cardinality of the relationship class. It was considered to create a many-to-many relationship class between PUK_Rooms and Cables. Some cables stretch through many rooms, and the sources' and endpoints' relationships with the rooms are already represented in the Room_has_Switches and Room_has_NP relationship classes. Many cables are laid along the same route, only to end in different parts of the same room. Creating a relationship between each of these cables and each of the rooms will create a sizable relationship class. When the size of the relationship class and the quantity of recurring data was considered opposed to the quality of output data, it was found that creating a relationship class between Cables and PUK_Rooms was not feasible. Cables_has_Switches is the only many-to-many relationship class and uses a intermediate table. Therefore the Object_ID of the Cables feature class serves as both a primary key and foreign key. The same is true for the Object_ID field of the Switch feature class.

3.6 Define the spatial properties of the dataset

Section 3.6 describes the spatial properties of the dataset in terms of selecting a coordinate system, defining topological rules and creating a network. This sections starts by comparing three coordinate systems and implementing the most suitable one. The section then compares a geometric network to a network dataset. Both of the networks' advantages and disadvantages are explored, after which one is defined as the network of the feature dataset. Finally the section will consider topological rules for the dataset, and provide a list of topology rules.

3.6.1 Spatial coordinate systems

The locally based datum Hartebeeshoek94 in South Africa is almost identical to the WGS84 (MMS, 2011). In terms of the scale of this project, the differences between the two datums are so small that it may be ignored. Choosing to employ either of the datums would have little effect on the outcome of the data model. It was decided to implement the WGS84, due to the fact that the spatial coordinate system of the referenced data (QuickBird satellite image) which was implemented was pre-defined to the WGS84 datum before the creation of this data model. In order to perform spatial analysis on the features in the network it was decided to employ a projected coordinate system. After comparing three projected coordinate systems in Chapter 2, it was decided to implement the UTM 35 South coordinate system (MMS, 2011; Wonnacott, 1999).

3.6.2 Topology rules

Topology rules are incorporated into a geodatabase to ensure that the data is spatially accurate. A topology is located in a feature dataset and contains a list of spatial rules, which the data needs to abide by. Only feature classes contained in the feature dataset are allowed to participate in the topology. ESRI's (2011) ArcGIS 10 software offers 32 topology rules, which ensures the spatial integrity when features interact with one another as well as

with themselves. The ArcGIS 10 software allows the designer to choose the topology rules which best fits the data model.

Topology rules are designed to simulate real-life limitations. Although the majority of projects do have limitations, some also have features which are exempt from certain rules. The ArcGIS 10 software offers flexibility by presenting an exception option, which gives the designer the opportunity to manually choose the features that are exempt from each rule (ESRI, 2011).

A limitation encountered with ArcGIS 10 topology rules, are that the topologies do not take z-values into account. This means that the GIS has a lack of three-dimensional topological rules. If two features have the same x- and y-values but different z-values, 3D environments recognize that these are two individual features, separated by a vertical height. A 2D environment ignores the z-values and believes that the features share the same physical space. In the same way, a topological rule ignores the z-values, and considers all features in terms of a 2D environment. This characteristic limits the diversity of the topological rules included in this data model (ESRI, 2011).

When deciding on topology rules for a data model it is very important to understand the real-life limitations being represented. A polygon feature such as a building, room or zone is not allowed to overlap itself. For example two buildings cannot be in the same physical space. It is also imperative that the Rooms polygon be covered by the buildings polygon. This ensures that all rooms are situated inside of buildings. Network ports serve as the end-points of the network and are only connected to cables. It is vital to create a topology rule which guarantees that all network ports are connected to the end-points of cables. In the same way, a switch is not allowed to be disconnected from the network. A topology rule must ensure that a switch is always connected to the end-point of at least one cable, and thus the network. Network ports and switches are both indoor hardware types which are not suitable for outdoor use. It is thus important that both of these point features always be situated inside of a building.

A number of topological rules were considered for the IT feature dataset topology, the first of which was the “must not overlap”-rule, as seen in Figure 3.15.

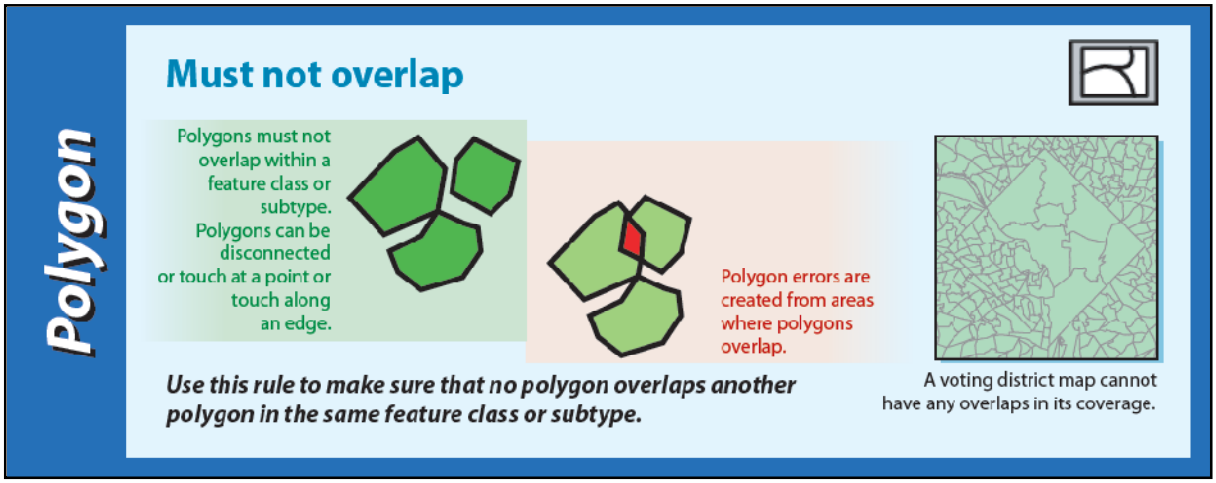

Figure 3.15: Must not overlap rule (ESRI, 2011)

This rule ensures that features from the same feature class do not overlap. The rule was incorporated for each of the PUK_Buildings and PUK_Zones feature classes. It was also considered for the PUK_Rooms feature class but could not be implemented. Due to the fact that the PUK_Rooms feature class has features with the same x- and y-values but differential z-values; the “must not overlap”-rule ignored the z-values and considered the features as errors. This caused virtually all the rooms to be considered errors (Figure 3.16). For this reason the “must not overlap”-rule was not employed for the PUK_Rooms feature class.

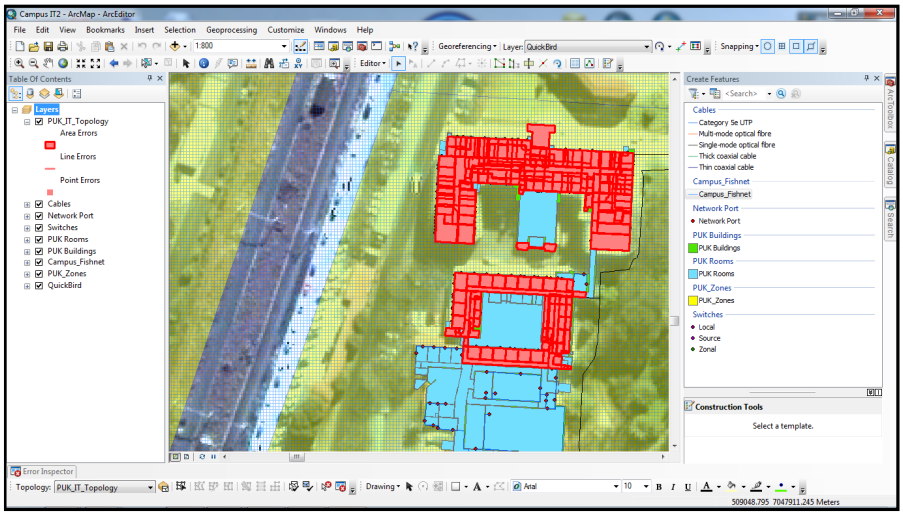

Figure 3.16: PUK_Rooms must not overlap error

To make sure that all PUK_Rooms were fully inside of buildings; the “must be covered”-rule was utilized. The “must be covered”-rule checks that a certain polygon feature class is entirely covered by another feature class, as seen in Figure 3.17.

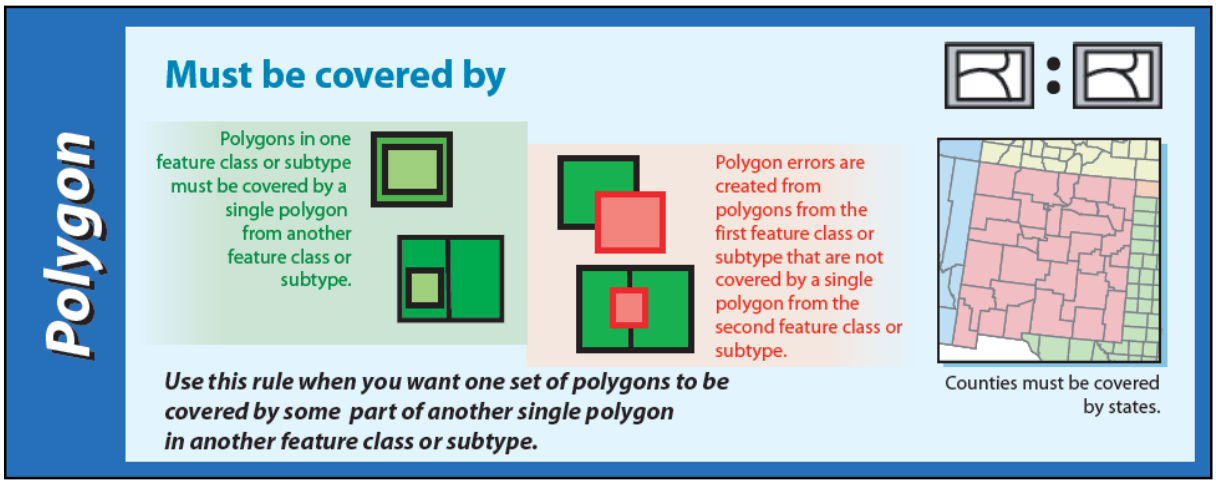

Figure 3.17: Must be covered by rule (ESRI, 2011)

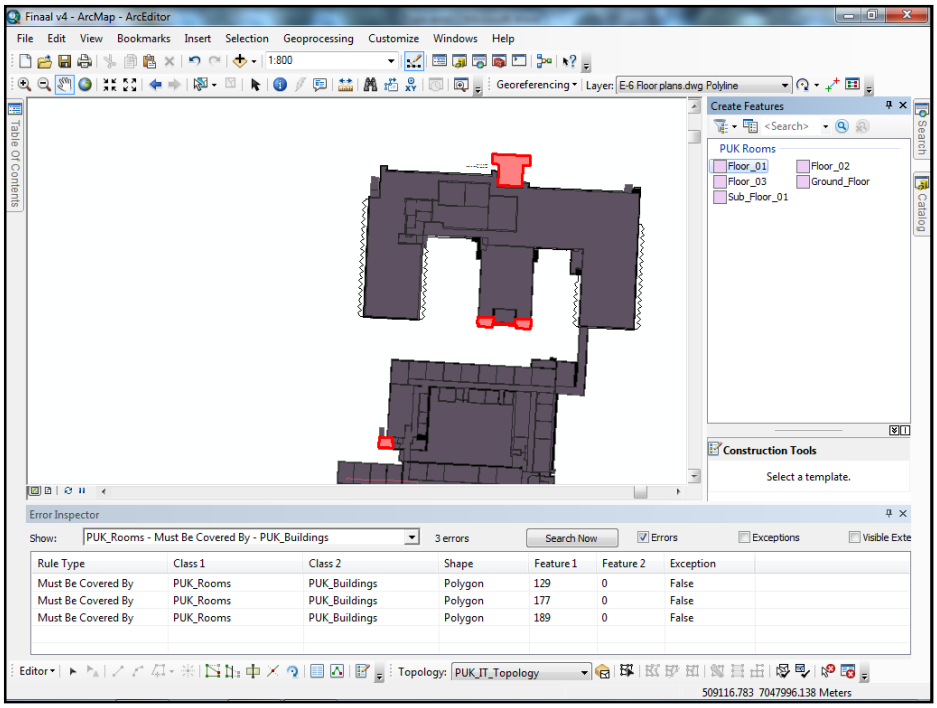

Figure 3.18: PUK_Rooms must be covered by PUK_Buildings error

In terms of this data model, the rule was employed in order to ensure that all the PUK_Rooms features were covered by PUK_Buildings. The rule worked excellently for most part of the data model. There was however certain areas where the topology rule failed to represent real-world limitations perfectly accurately. Some external areas share a boundary

with the building. Although the external areas did not exceed the building border line, it was not totally covered by the border, and thus seen as an error (Figure 3.18).

This obstacle highlights a limitation in the GIS topology rules. A rule is needed which ensures that a polygon feature class is covered by another polygon feature class, which includes the boundaries. In order to overcome this obstacle and because this incident only appeared on three occasions, it was decided to define the three incidents as exceptions to the rule, because external areas are seen as part of the building. If the data model is applied to the whole campus, the number of the external area errors will increase to some extent. The external area errors for the rest of the campus can also be marked as exceptions.

Network ports as well as switches play vital roles in the network, but have no function if not connected to it. It is thus essential to employ a rule which guarantees that all network ports and switches are connected to the end-points of cables and consequently the network. The topology rule used to ensure this was the “must be covered by endpoint of”-rule (Figure 3.19).

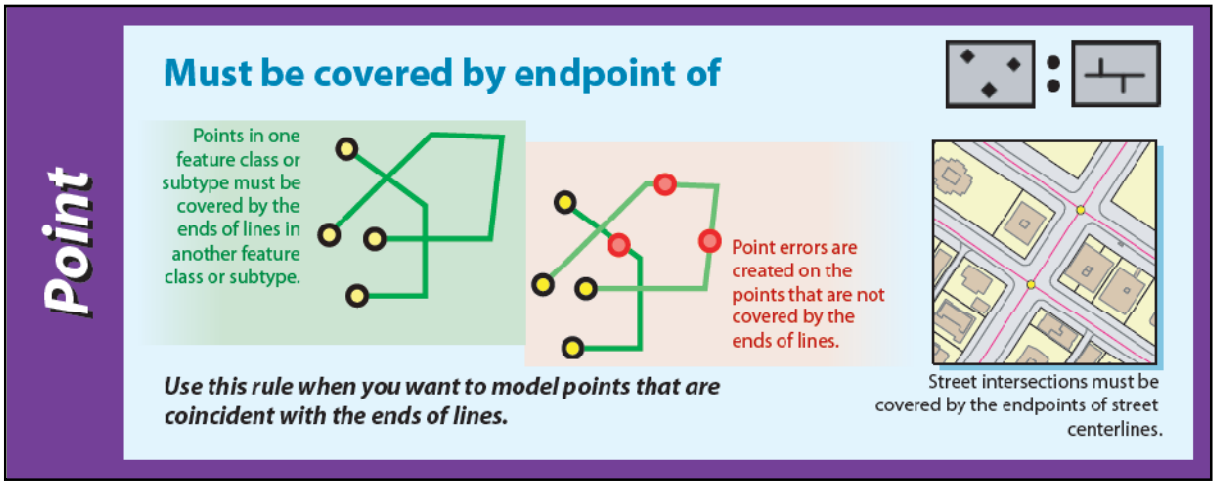

Figure 3.19: Must be covered by endpoint of rule (ESRI, 2011)

This rule is applicable only for point feature classes (Network_Port and Switches). It ensures that all of the specified points are located on the endpoint of a feature of a specified line feature class. Essentially this rule depicts all stand-alone switches and network ports as topological errors.

The penultimate topology rule which was considered was a rule which makes sure that a line does not intersect itself. This rule was considered because a network cable cannot intersect itself without connecting to a switch first. The rule considered was the “must not self intersect”-rule, as depicted in Figure 3.20.

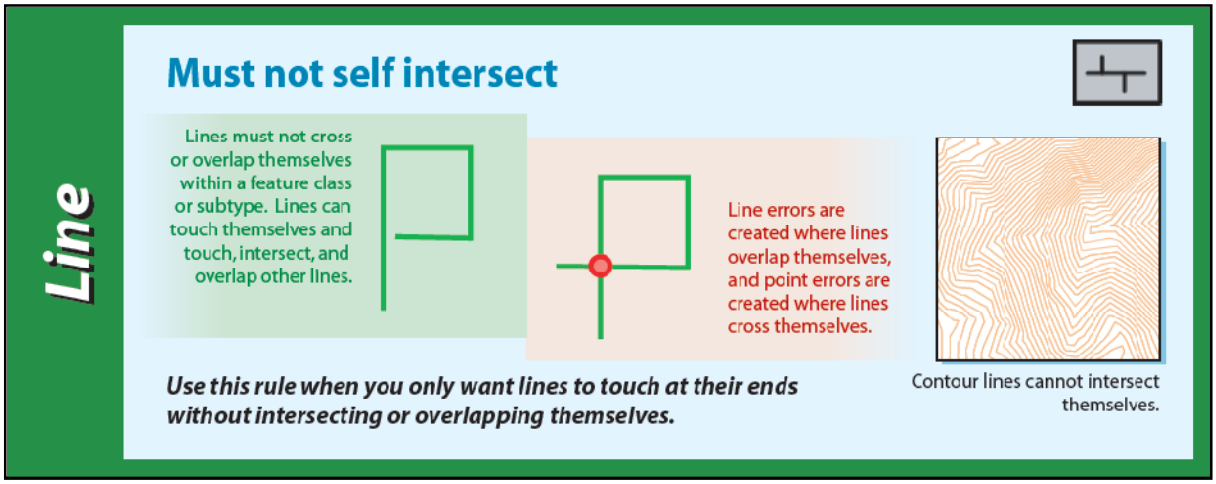

Figure 3.20: Must not self intersect rule (ESRI, 2011)

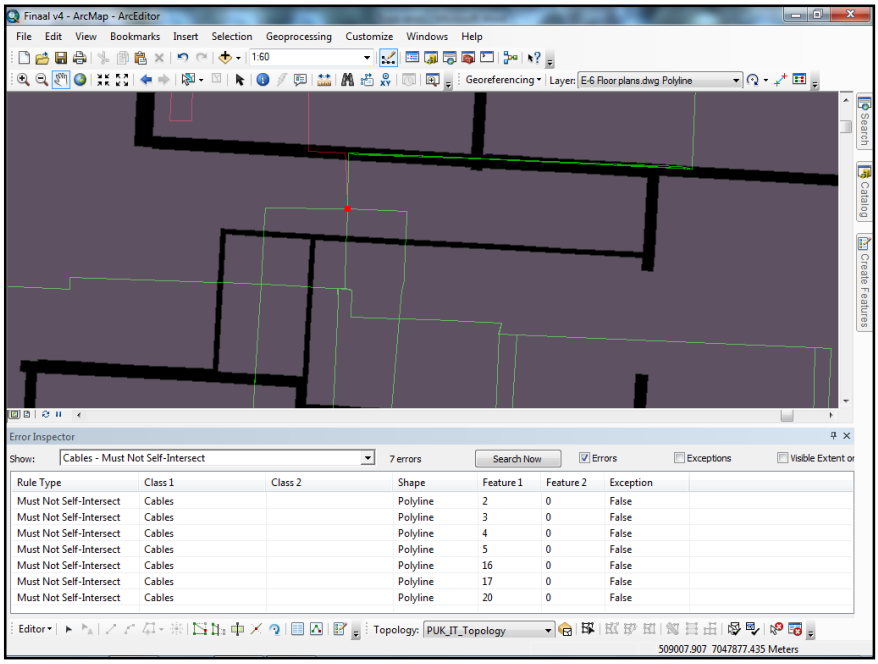

Figure 3.21: Cables must not self intersect error

This rule proved to be problematic in terms of detecting the z-values. As the software ignores the z-value, it considers all the lines to be on the same height level. Consequently, cables which are on higher levels and not spatially near cables on a lower level are

considered intersection errors. In the same way vertical lines are depicted in a 2D environment as points. A vertical line as a result is considered to intersect any other lines which share the same x- and y-values, as well as itself (Figure 3.21). The “must not self intersect”-rule was found to be not feasible for this data model and therefore excluded.

The final topology rule considered was the “must be properly inside polygons”-rule. This rule ensures that the features of a certain point feature class are covered by a specified polygon feature class (Figure 3.22).

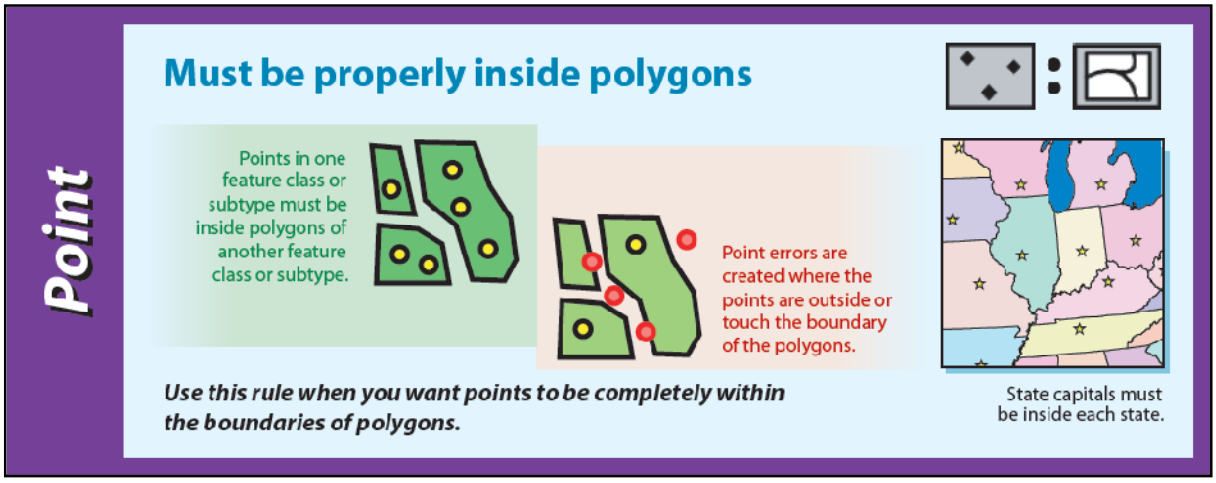

Figure 3.22: Must be properly inside polygons rule (ESRI, 2011)

Network ports and switches can only function indoors, and are only allowed to be located inside a building. This rule was incorporated in order to ensure that both the Network_Port features as well as the Switches features were located inside PUK_Buildings.

A total of seven topological rules were created for the IT feature dataset (Table 3.11). Overlooking the z-values is a serious flaw in the software which leads to an absence of three dimensional topologies. Two dimensional topologies are however of use and creates a list of rules which ensures that real-life limitations are accurately represented. Although some topology rules were omitted from the topology; it still ensures that some data integrity is kept intact.

Topology rules		
Primary feature class	**Topology rule**	**Secondary feature class**
PUK_Zones	must not overlap	None
PUK_Buildings	must not overlap	None
PUK_Rooms	must be covered by	PUK_Buildings
Network_Ports	must be covered by endpoint of	Cables
Switches	must be covered by endpoint of	Cables
Network_Ports	must be properly inside	PUK_Buildings
Switches	must be properly inside	PUK_Buildings

Table 3.11: Topology rules for the IT feature dataset

3.6.3 Network design

A network is a type of topology, which ensures that features are connected spatially. There are two types of networks in GIS: a geometric network and a network dataset. A geometric network is normally used for utility features or features which travel in a single direction along a network, such as an electrical network, or a river network. A geometric network can perform network analysis on the utility network. ArcMap implements the Utility Network Analyst toolbar to perform network analysis such as downstream tracing; upstream tracing; find a common ancestor; and find the shortest path between two points (ESRI, 2011).

A network dataset is mostly implemented to display the connectivity of a network, which displays connectivity between features in both directions along a network. Normally a network dataset is used to display a transportation network for streets, railways or bus routes. The main difference between a network dataset and a geometric network is the analysis methods that they offer. A geometric network places the analysis focus on analyzing either upstream or downstream, whereas a network dataset analysis the network in terms of routes in any direction. In ArcMap the Network Analyst toolbar is used to perform network analysis on a network dataset such as creating a route layer; define service areas around a certain point; as well as find the closest facility to a certain point (ESRI, 2011).

In normal circumstances a geometric network would be suitable to represent a utility network such as IT facilities. It offers the right analysis tools as well as the correct hierarchical layout of the network. However, the aim of this study remains to determine to what extent GIS software can be implemented in order to manage, analyze and visually illustrate an IT-network between buildings as well as inside of buildings on a campus. Modeling the indoor IT-utility infrastructure for a building needs to be performed in a 3D environment (ESRI, 2011). According to van Maren (2011), geometric networks are not yet supported in a three dimensional environment by the latest ArcGIS 10 software. The person goes on to state that 3D geometric networks are a focus area for the next software release and that network datasets are however supported by ArcGIS 10 in 3D.

It was decided to implement a network dataset, instead of a geometric network, to represent the network connectivity of the data model. Although the analysis would be limited by implementing a network dataset; some network analysis would however be possible in 3D. Only the analysis method for finding the best route could be implemented on the IT infrastructure. The “finding the best route” tool can be implemented in such a way, that it imitates certain geometric network analysis methods. The analysis methods are described in more detail in Chapter 4.

3.7 Propose a geodatabase design

The seventh step in Actur & Zeiler’s (2004) geodatabase design method is to propose a database design. The proposed design is sub-divided into two categories, both of which will be represented visually. The first part is a description of the suggested design’s layout (Figure 3.23). The data model firstly employs a file geodatabase (PUK_Geodatabase) as the central storage point for all the spatial and non-spatial data. The first object contained in the geodatabase is the PUK_IT feature dataset, which in turn contains all of the feature classes. The PUK_IT feature dataset also contain the topology (PUK_IT_Topology) as well as the network dataset (PUK_IT_ND).

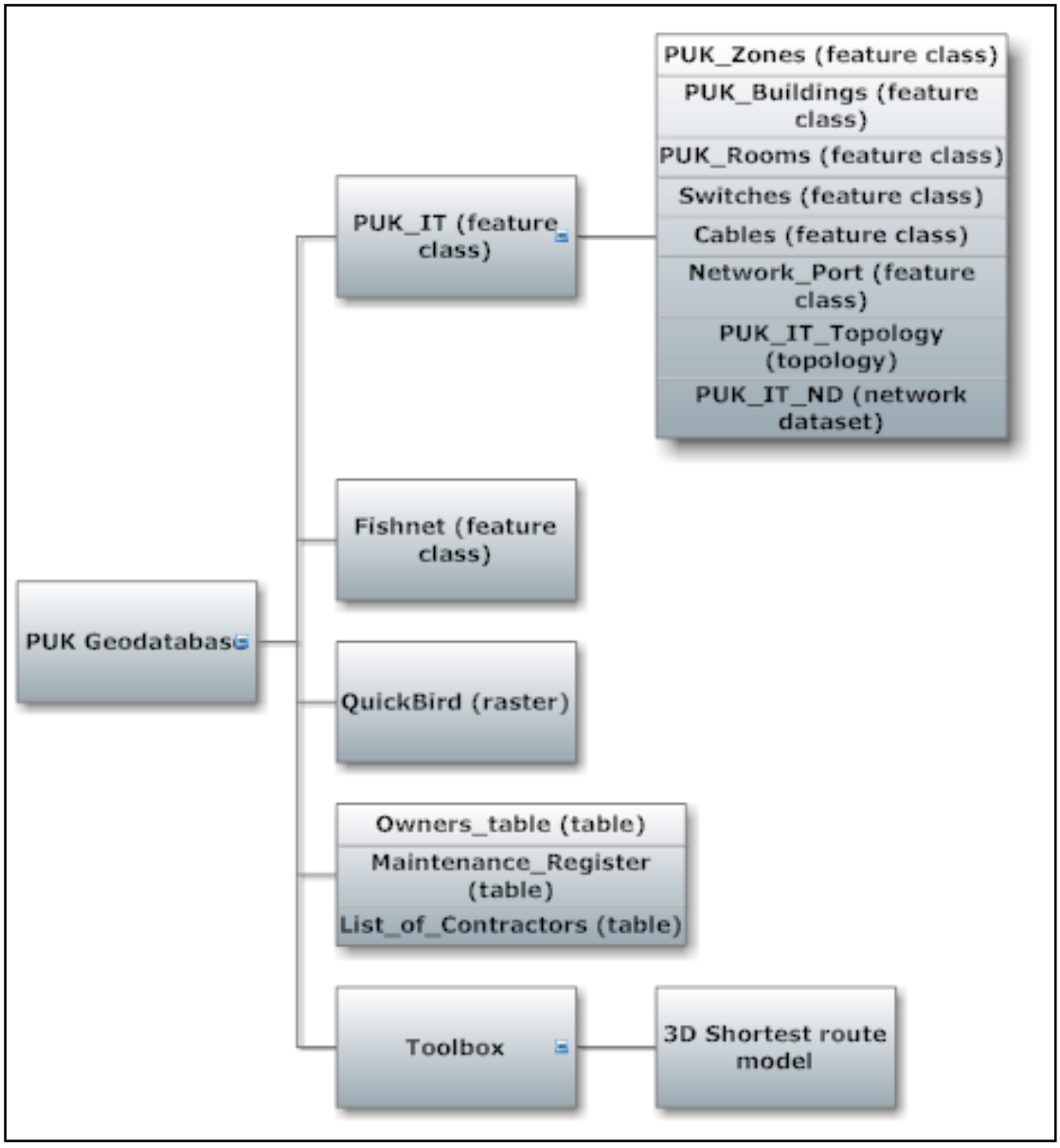

Figure 3.23: Layout of the PUK_Geodatabase

The three non-spatial tables are stored and managed directly in the PUK_Geodatabase. The Fishnet and QuickBird rasters are also stored directly in the geodatabase. Finally the geodatabase contains a Toolbox. A toolbox is a storage location for models. This particular model, the 3D Shortest route model determines the shortest route between two or more points along a network.

The second part of the geodatabase design description entails visually describing the relationships among the features in the PUK_IT feature dataset. The relationship classes are depicted in Figure 3.24 by the arrowed lines between the features. An arrowed line with only

one arrow, describes a one-to-many relationship. The line points away from the origin table towards the destination table, and contains the text: “has many”. The first example in Figure 3.24 reads: “PUK_Zones has many PUK_Buildings”. In the same way; a double arrowed line which reads “have many”, represents a many-to-many relationship. A double headed arrow which reads “has one” describes a one-to-one relationship class.

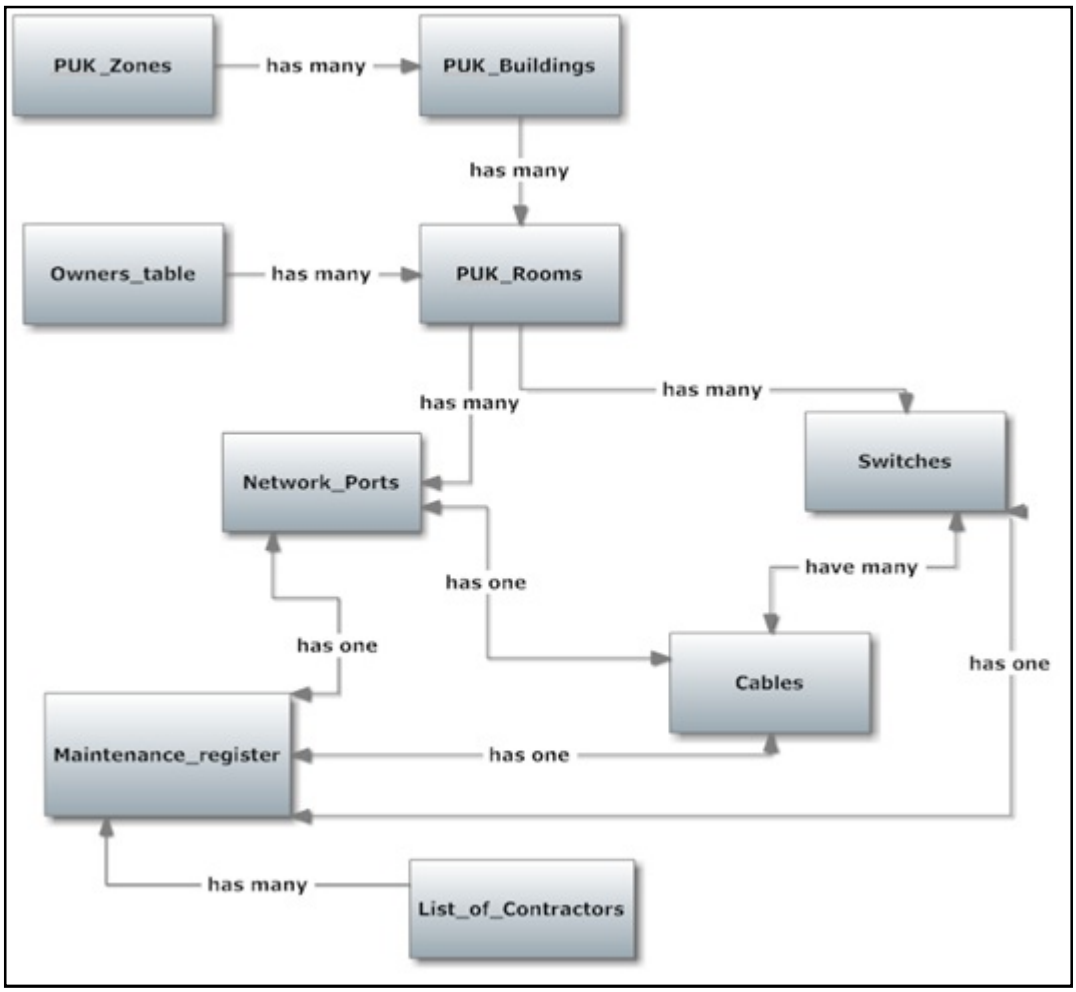

Figure 3.24: PUK_Geodatabase relationship classes

Step 7 is a summarization of all the logical design steps. It creates a visual outline of the collective datasets, feature classes, rasters, domains, attributes, tables, networks, topologies and subtypes which were described by steps 4 through 6. Step 7 sees the end of the logical design steps as well as the end of Chapter 3; the following step is the initiation of the physical design phase.

Chapter 4 describes steps eight through ten of the Actur and Zeiler (2004) model. This chapter illustrates the different implementation techniques which were compared to one another in order to develop this model.