Model Base Systems Engineering used in
developing a Telescope Manager

Gerhard le Roux
Senior Systems Engineer TM Consortium (SKA Africa)
gerhard@ska.ac.za

Abstract. Various tools, tutorials and examples exist about the idea of doing your development based on models of information (MBSE) as opposed to the traditional document driven approach. Yet, the practical examples of achieving this are still relatively few. An unguided modelling approach can easily lead to "modelling for modelling’s sake", paying for expensive management tools - and creating diagrams that makes the project more complex instead of simpler.

This paper is a case study on the efforts to move to a model based development for a large complex system by looking at early conceptual and preliminary design stages of the Telescope Manager project - particularly taking into consideration how it transforms from a functional to a physical design.

The aim is to see if general principles and concepts can be derived upon which future systems engineers can make use of before embarking on a similar venture.

The study will define the problem area and scope of modelling required for the Telescope manager.

Then the study will give a critical examination of various strategies and actions undertaken regarding the use of MBSE for the Telescope Manager.

The study will conclude with a derived set of concepts and principles based on the evaluation results that may be used in the future.

Lastly the study will comment on what the future looks like for MBSE and give some recommendations in terms of preparing for what comes next.

Introduction and Background

The required system. The Telescope Manager (TM) will be a subsystem of each of the two SKA telescopes under development. These telescopes are arrays consisting of a number of spatially separated antennae that observe the universe in the radio wave spectrum (50 MHz to 15GHz) based on radio interferometry and synthetic beam forming. In particular the two Telescopes are named the SKA Low (frequencies up to 350 Mhz and based on dipole antennae) and the SKA Mid (frequencies up to 15Ghz and composed of 15m Dishes).

As a system an SKA Telescope can be characterised as having both enormous size (in terms of number of elements) as well as functional complexity (in terms of interrelated tasks and actions). The control of the telescope is therefore hierarchical with processing distributed both functionally and spatially. At the top level the functions are managed by means of abstract
management and monitoring of the lower level capabilities and configurations of the Telescope. This is the purpose of the Telescope Manager. Even with the hierarchical levels and abstraction implemented, the Telescope manager will have to interface with a very high number of components and coordinate the correct execution of scientific observations (see Figure 1 and Figure 2). In addition, it needs to assist an operator of the telescope by presenting the state of the system in such a manner so that a human person can have a high level of awareness of what is going on with the instrument.
The project. A global consortium with members distributed across 7 countries executes the development of this system. Communication is therefore one of the main challenges.

The TM itself have various layers of hierarchy to reduce its complexity and is broken down into Sub Elements which in turn is broken down into Applications – which in turn is broken down into components. Each of these item will, during its life cycle, produce information that cover at least: (1) its requirements (2) its architecture (3) its external interfaces. During the production and utilisation stages the information will change to cover at least: (1) its verification and validation (2) its physical and computer code specifications and (3) its operational use and support descriptions.

It is vital that this large body of information be organised, maintained and communicated effectively and efficiently during the entire life cycle of the TM system. This is more than just a configuration management challenge. The design is information. The quantity and complexity of information makes information technology a key driver to ensuring integrity and coherency of the design.

The enabling system. It is therefore clear that information technology, as an enabling system will play a key role in successfully developing the Telescope Manager. The consortium therefore made the following decisions early in the project:

1. That as much as possible of the design descriptions be done using SysML as the language.
2. That requirements be managed as integrated elements of a database as apposed to free standing documents.
3. That the three main aspects of the system (its requirements, its configuration or interfaces, and its behaviour) be managed as an integrated set of elements that can produce a coherent description of the system at a particular stage.

To this end, a modelling tool based on SysML language with cloud base collaboration features was purchased as the underlying technology for implementing such an enabling system. Its goal was to use Model Based Systems Engineering (MBSE) processes to more effectively and efficiently manage the development of the system.

MBSE. The recent trend to view the design description of a system over its life cycle as a model of information as apposed to a collection of documents is referred to as Model Based Systems Engineering (MBSE). In other words “the primary artefact of those activities is an integrated, coherent, and consistent system model, created by using dedicated systems modeling tools” (Delligatti, 2014, p. 3). This new paradigm posits that the information is essentially a structured model that describes the information as data with relationships to other data. The underlying model therefore is elevated in importance and used to address all the aspects of engineering (specification, design, integration, validation, and operation of a system). How these data are structured is partly imposed by (1) the formal rules of the modelling language language (SysML), (2) the model of the required system, (3) the architectural concepts and principles used and (4) the life cycle model used to view the development of the system (Jeff, 2008).

The practical implementation of this can however be challenging. For example the tool selected as the underlying technology, have its own idiosyncrasies of approach regarding (1) how to structure the project, (3) how to manage presentation of information and (4) how to manage
changes to information. The processes used by the developing team have to try and fit to the processes assumed will be used by the tool. Despite the rapid growth in information technology, the expected increase in productivity for large complex systems engineering projects is disappointing (Brynjolfsson, 1993). Often, the perception (albeit somewhat subjectively) is that MBSE processes are mostly at the level of either (1) “we don’t know what we are doing and it doesn’t work” or (2) “we don’t know what we are doing but it seems to work” (Burch, 1970).

Enterprises opting to implement a MBSE process therefore run the risk of consuming large amounts of effort with a low likelihood of success.

There is therefore a need for a body of knowledge that gives both general principles and concepts as well as practical guidelines for implementing a MBSE process in a company.

This paper aims to document experiences of trying to implement a MBSE approach with the hope that the lessons learned so far can be used as points of departure for further studies that can advance the knowledge regarding the use of MBSE.

Definitions of concepts used in this document.

1. Model: As abstract description of a system under development based on a formal language.
2. Model Database: The repository that stores all the modeling information in a structured manner so that a Tool can retrieve, create or update the information.
3. MBSE Tool: A tool that allows a user to create and manage a model for a system based on SysML language.
4. Auxiliary Tools: A tool that allows a user to convert information captured in non SysML languages (i.e. documents, spreadsheets etc.) in a predefined manner so that it can be imported by an MBSE tool and become part of the Model.
5. Document: A structured layout and presentation of information by means of human readable text and image objects. A Document can be implemented by means of print on paper, web pages or digital paper (e.g. pdfs).

Strategies and actions used to Implement MBSE for TM

Need for a common approach across diverse cultures. Different cultures bring in different behaviour, shared attitudes, and common expectations in terms of interpreting and solving a problem (Novinger, 2001). Because the consortium for developing the TM is constituted out of a diverse group of countries, it was pertinent that a means be developed to standardize on a number of processes and tools.

The first decision that needed to be made was which modelling language should be used. The TM consortium choose SysML mostly due to external reasons:

1. It was relatively well known across the globe;
2. Being based on UML concepts, the consortium (who had a large contingency in software engineering) should be able to learn the language easily;
3. But most importantly, a large part of other consortiums in the SKA project (including the SKA office) choose this is their modelling language.

Therefore, other than the fact that the TM project was expected to involve a lot of software engineering, which are used to being modelled in UML, the selection of SysML was not based on the characteristics of TM itself. However by having a standard language used by all
members for modelling, the TM consortium would be able to communicate difficult concepts across the group unambiguously.

The next challenge was to ensure everyone understood the language. On one level this went quickly, the basics and meanings of SysML was quickly learned by key members, but a much steeper learning curve was needed when more particular aspects idiosyncratic to the project needed description.

One example was the need to model interfaces. A large part of TM is the implementation of connections with other elements of the Telescope by means of Interface Control Documents (ICD’s). The ICD’s are agreements between the two connecting parties (the third being SKA office as the authority responsible for designing the Telescope) on (1) what functions will be exposed over their interfaces, (2) how it will be accessed and (3) all the implementation aspects of these functions that affect both parties. Having a standard modelling language in which the ICD’s can be described unambiguously can bring a number of advantageous. SysML doesn’t restrict one in terms of how to model interfaces but it does require (1) deep understanding of SysML and (2) the context of the SKA project to convey the information in such a manner that it reflects the nature of the intended relationship appropriately. Some examples of challenges may be

1. Modelling an interface that makes use of an external network to implement the connection.
2. Modelling an interface that is physically a number of distinct services (containment, installation, deployment etc.) provided by infrastructure as an abstract functional relationship.

Up till now, the ICD Document itself (not the model) remained the regulating instrument.

A second example of deep insight required before proceeding was how to tailor the language for a particular system such as the Telescope manager within the context of SKA. By tailoring here is mostly implied the use of sub types (called stereotypes in SysML) of standard language types defined in the language. For example defining a stereotype Product System (<<Product System>>) to indicate the block is a specific type of system within the context of SKA. Different cultures have different “mental models” (Johnson-Laird, 1983) of a problem; and while a language restrict the manner in which the mental models can differ, defining a new “sub language” or “meta-model” is even more difficult if participants have different interpretations. Yet ensuring a domain language is appropriately modelled and communicated can have far reaching effects. So a deep insight of the principles of SysML modelling as well as proper domain knowledge of the SKA Telescope had to be gained over time before a good tailored modelling language could be defined.

**Communicating information versus defining information.** The information generated during development had to be reviewed, changed and approved before being used as input information to downstream development processes. A critical aspect to a MBSE processes is its ability to communicate information. By communication is meant the ability to create a shared meaning across different human stakeholders (Ting-Toomey, 1999).

The TM consortium started by using Google doc’s technology to present and edit information in an informal environment. The Google doc’s technology allowed information to be created and reviewed in a collaborative manner thus creating an environment whereby different members can communicate information that is recorded, traceable and changeable to a
distributed set of members. The key driver for this approach was speed of collaboration as opposed to having an underlying model based structure. Google documents could be viewed quickly, changed quickly, commented and responded to quickly.

However, although the information was structured (chapters, sections, paragraphs, tables, pictures etc.) ultimately the Document was just a view on an underlying model that should be depicted by means of multiple views.

As an example consider the TM systems specification Document. This Document generated a set of requirements scoped and sectioned according to the standard system specification layout. Each requirement however had various attributes that could allow for different views of the system. For example the relationship between systems requirements and subsystems requirements (parts of the system) was shown by an attribute linking the derived requirement to the parent requirement as well as an attribute that linked the parent requirement to the subsystem. Thus a view indicating subsystem with corresponding source requirements could be generated. The same will hold for attributes describing risk, maturity, dependencies, review status, rationale and assumptions. It was therefore decided to try and incorporate key Documents (requirements and design architecture) into a Model Database.

At this juncture a large set of information had to be transformed into model-compatible elements before being imported into the modelling tool. The consortium used scripts that could parse the elements out of Documents based on how it is structured and presented in the Document. For example each requirement in the Document had a unique ID starting with a descriptor code that were, if not unique, highly unlikely to be used anywhere else in the Document. These “proxy’s” could then be used as a logical search pattern for a script to load the succeeding data into a database. The scripts however had to be clever enough to interpret different character presentations, paragraphs styles, numbering schemes etc. In summary: to be able to re-interpret the data presented in a Document, a deep insight into the underlying model of a Document is necessary.

After information was successfully incorporated into the Model Database, the challenges shifted to being able to modify, edit and present the information. It soon became clear that after the Document was sliced up and loaded into a database, it was vital that the information in the model be glued together again to form various interpretations in Document format. If the Document was now not anymore the primary means of capturing and displaying information it still remained a very important asset in communicating information – especially when you have a diverse and distributed group of stakeholders.

Therefore the scripts needed to be modified to also generate Documents. However since both functions (generating and extraction) of scripts are based on an underlying Document model, the knowledge required for generation could be based on what was learned in extracting.

It should be noted here that various MBSE tools come with features to generate Documents based on what is stated inside the model. This was considered as an option but was finally not used for the following reasons:

1. In most cases the tool instead of the creator of the Document predetermined the structure of how the information should be displayed (which chapters should contain which requirements and which requirements should come before others).
2. In most cases the tool instead of the creator of the Document predetermined how elements inside the model should be interpreted into Document types of information (Requirement into new paragraph or table or list).

The different transformations of data during the life cycle stages. When looking at available examples of how SysML models the design of a system it is often missed that in practice the modeller doesn’t have all the information at hand when producing the model. What happens in practice is that information is added piece-wisely as the system progresses along its life cycle. The information generated changes not only in scope and fidelity but also in type and character (see Figure 3). As an example, during the conceptual stages the “things” that were being discussed were the OBSMGT and TELMGT system (both sub systems of TM). These were abstract functional domains without any reference to how they will function internally and where they will be deployed physically. All that was known was that each would concentrate on a specialising field of managing a Telescope: Observation management and Telescope Management. As the project moved to a more concrete design stage, the emphasis needed to shift from thinking purely in abstract problem domains to actual entities performing services and providing capabilities. Now the focus has shifted to the applications (which can still logically be grouped under OBSMGT or TELMGT) with a more definite usage of computer resources and a thus a better indication of deployment needs. These applications are however still thought of as pure functional entities in the sense that their description is defined in terms of their purpose within the larger system. It is expected though that a further evolution will turn these applications into actual physical products (i.e. something that can be affected and constrained by the physical environment); that are either reused, or developed for the specific purpose (an office communication and workflow organizing tool later gets called simply Microsoft Office).

Figure 3: Life cycle model of information

The challenge for a MBSE tool was to create various descriptions of the system that corresponds to the different life cycle stages of the system whilst still maintaining the necessary level of integration between the models. If each stage produced information that becomes
useless after being used as input information, then describing and modelling the relationships between the information at the different stages gain little value. On the other hand if changes to a system causes an impact on information stated upstream and or downstream, then maintaining traceability between these will be very valuable. For TM three stages of information was defined (see Figure 4):

1. The problem domain stage: Information is presented as “sketches” that iterates, evolves or become extinct. The presentation should enhance logical reasoning and explicate complex concepts by depicting key information aspects unambiguously. However the cost of producing, changing and presentation should be low. The maintenance and management constraints on information should be low because of its high likelihood of becoming extinct, replaced, or transformed. Information captured inside the problem domain stage is not coupled with information in the other stages.

2. The functional configuration stage: Information is placed in a structured Model based on a formal language that fits the problem domain. The information is structured according to abstract functional relationships and attributes as apposed to physical, spatial and geographical constraints. SysML language with appropriate domain language extensions is used to formally capture the information. The changes of information are formally managed by a tool that can appropriately capture the required changes across all information elements in order for the model to remain logically coherent.

3. The physical configuration stage: Information is still placed in a structured model but the information elements are depicted and modelled in the physical domain as opposed to the functional domain. The physical domain can be represented by a number of physical model environments (Finite Element Model, Electrical Circuit Models, Chemical Reaction Model etc.). Functional configuration information should be coupled with information in the physical domain by means of a gateway mechanism that separates the layer between functional and physical. The management of changes to information thus becomes layered by separating changes into physical impact and functional impact. Thus: physical changes will be managed by the physical model, whilst functional changes will be managed by the functional model. A gateway mechanism can be a configuration management tool that purely manages informational relationships between “pointers” to physical and functional entities (e.g. this “item” is represented physically by these “items”).

Figure 4: Generic life cycle of information
The relation of requirements and requirements management to the model. A key part of the system’s Model is its requirements and their relationship to other elements of the model. The last part “relationship with other elements” is what makes it necessary to integrate the requirements into the model. SysML caters for the description of requirements as part of its language and thus allows tools to incorporate the requirements into the Model.

Unfortunately most application tools specialise in either requirements management or systems modelling. The performance of a MBSE tool that incorporates requirements with the Model can become constrained if the number of requirements to be modelled becomes large.

As mentioned earlier (see “Communicating information versus defining information”) the Document still remains an important mechanism to communicate requirements. It was also found that the existence of a requirement outside of a Document becomes difficult to interpret. Good well-written requirements should always be able to stand on its own feed. However when you remove the context of the requirement, (where in the Document it is stated) you bring a in a lot of confusion, misinterpretation and lack of clarity regarding the requirement. A possible reason for this maybe how the human mind process information cognitively. Humans have a limited set of items (5 -7) that can be stored in short-term memory (Miller, 1956). If you show a person therefore a spread-sheet of requirements instead of contextualised requirements (nested into related topics and chapters) the human mind becomes preoccupied with storing and retrieving information instead of determining meaning from what is being written. It became clear that the position of a requirement relative to other information should become part of the systems model so that requirements can be structured in the correct context when presented to a human.

Observations and Evaluations

Some feedback from users of Tool and language. A common frustration observed by TM members of the MBSE tool was that often you created more information elements than was necessary for the specific use. For example a user wanted simply to explain that one element has a relationship with another. For the model to remain logical coherent, a number of attributes and characteristics needs to be created as additional information elements. Having a separation between problem domain information and other stages later mitigated this problem. Therefore a user opted for a sketch when simply trying to reason about something then capturing it immediately in the MBSE tool. A similar problem observed by members was that in some cases depicting something by means of a SysML took more effort than depicting it in natural language. Again by separating information into sketched versus formal configuration this problem was mitigated. In addition, it was also decided later that a user could (and should) enrich the model by having descriptions in words as distinct model elements to compliment the information captured.

A secondary problem was the effort required to import and export large quantities of information within and without the tool. Without the ability to work with large quantities of information the tool become worthless for complex large projects. Without the ability to present information in formats that fits the unique properties of an organisation and project, large pieces of shared meaning is lost. For TM the decision was made to spend significant amounts of resources on creating such a capability for an enabling system. Most tools do come with add-ons to support this ability. However, a shift in focus may be necessary - the exporting and importing of information is not an auxiliary function but a primary function.
The tyranny of the Document
It is often said that the MBSE will not replace the Document. The written word has become a ubiquitous mechanism for allowing communication between humans by essentially removing natural constraints (such as a human mind’s capacity to memorize or being at the same place and time when information is expressed). In most large complex projects the number of formal documents with multiple authors and multiple signatures can easily reach a thousand. The same situation was found for TM development. Despite efforts to capture information in a Model the documents still increased rapidly over time.

However, upon closer inspection a different picture emerges. The requirement for large quantities of documents is a reflection of (1) a client needing information as intermediate deliverables of work effort, (2) separate teams requiring input information to complete tasks and (3) descriptions of the system is necessary to ensure changes can be investigated and implemented on a rational and logical basis. But the most important fact is that the need is not for documents but for information. Documents are simply a presentation of information. Therefore, if the information is captured in a model instead of independent documents, it is the model that instantiates representations of information in the form of a document and not the document itself. The number of documents that can be created can therefore increase in scale without placing as much of a burden on the generation of the underlying information. To conclude therefore, the information demanded by large-scale projects is very large and will remain large; however, having an underlying model of the information should decrease the demands placed on managing documents, design data and relationships between the data.

Findings and recommendations

A life cycle model of information. It was found that organizing information in terms of a life cycle greatly reduced complexity in managing the information. For TM this was organized into a problem domain stage, a functional configuration stage and a physical configuration stage (see Figure 4). It is up to the reader how he wants to define and tailor the stages. The key principles regarding setting up a life cycle model for information are:

1. There is a fundamental dependency between stages that determines the effect of changes made to information in a stage. This is the main use of life cycle stages in that it allows a person to analyze the effect of changes in terms of where the change needs to be made (which stage contains the data?) and what will be affected by it (which stages are dependent on it). Note though that it doesn’t necessarily imply an exact time sequence between stages must be followed (first stage A then stage B) - one can iterate between stages and still maintain the dependency rules between the two stages.

2. There should be an underlying shift in the objectives and problems being tackled (aspects of concern) in between the stages; otherwise the two stages are really just the same stage. By just arbitrarily creating separate stages, one creates the risk of having a high level of coupling in-between the stages as well as creating duplicate sets of information that both describes the underlying state of the system.

Make generating and capturing information key capabilities of your enabling system. In large projects with large sets of information, users can spend significant time capturing information and generating information in appropriate Documents for communication. Therefore having tools that can take out the manual repetitive aspects of these processes can save time and money.

Key aspects of a good MBSE tool. In summary therefore when evaluating MBSE tools the following attributes were deemed valuable based on our experience:
1. Modelling both requirements information as well as system configuration information.
2. Being able to generate and present information into different formats (word documents, pdf’s, websites etc.) that is customizable and configurable.
3. Allow representing information according to different life cycle stages of the information.

**Conclusion**

**The value of MBSE.** In theory MBSE can alleviate drain on resources and waste in hours spend managing information of large-scale projects. It also can be a dangerous concept by setting the impression that it can be a bought out enabling system without additional resources needed. Our experience indicated that considerable hours needed to be spend in correct configuration of processes and systems to ensure it can deliver its desired effect. Acquiring an MBSE system needs to be managed in just the same manner as acquiring any complex system. Although underling information technology is rapidly growing, it is extremely unlikely that a MBSE tool will solve all information management problems, and the configuration and implementation of your system should trade off the desired features against the cost of implementing them.

**A vision of the future.** It is impossible to predict whether and when MBSE will replace more traditional information management systems. However there are some important technological trends that may impact these outcomes:

1. Implementing database services have become decoupled from enterprise management systems. Many database management systems can be deployed and used with minimal costs on web-based platforms. If the underling information model is really just a data schema for a database, then a large part of MBSE can be implemented on generic database systems. This can produce disruptive effects for Tool vendors having architected the information database as part of the tool.
2. Web application technology and more reliable and fast connection bandwidths have made the front ends of web applications richer and with more complex features. Although online drawing programs are still far away from CAD and other modelling tools deployed on desktops, the trend is moving towards doing graphical design on a web browser and thus providing the software as a service over the internet as opposed to a physical product (Choudhary, 2007). This again could bring disruptive effects to Tool vendors that architected the front end user interface of tools to be closely coupled with the underlying model domain.

This seems to indicate that in Future, MBSE tools will be divided up into underlying data management tools (databases, web servers and data model applications) and presentation and information capturing tools (drawing tools, spread-sheet tools and word editing tools). For example one could have a Google doc tool that represent information sourced from a separate database server that allows a user to formulate and present information in a standard document layout. One can also have a drawing tool that allows a user to graphically depict and display information based on an underling data model captured in a separate database.

All this should reduce costs of purchasing large-scale MBSE tools but it will move the effort of creating a MBSE system to the user itself. The recent advances in web technology may reward those who will be willing to spend more on man-hours instead of capital expenditure.
References


Biography

Gerhard le Roux has been doing systems engineering work in the South African Industry for 12 years. He has touched aspects of systems spanning small-scale theme rides for Gold Reef City to managing weapon systems for Ground Based Air Defence. He has worked in industries covering military defence, railway transportation and is now involved with radio interferometry telescopes. Gerhard has obtained a B.Eng. in Electrical & Electronic engineering, a B.Sc. in computer science and a Masters in Business Leadership. He is also an INCOSE Certified Systems Engineer.