Model Based Systems Engineering for Astronomical Projects

R. Karban, L. Andolfato. P. Bristow, G. Chiozzi*, M. Esselborn,
M. Schilling, C. Schmid, H. Sommer, M. Zamparelli
European Southern Observatory, Karl-Schwarzschild-Str. 2, 85748 Garching, Germany

ABSTRACT

Model Based Systems Engineering (MBSE) is an emerging field of systems engineering for which the System Modeling Language (SysML) is a key enabler for descriptive, prescriptive and predictive models. This paper surveys some of the capabilities, expectations and peculiarities of tools-assisted MBSE experienced in real-life astronomical projects. The examples range in depth and scope across a wide spectrum of applications (for example documentation, requirements, analysis, trade studies) and purposes (addressing a particular development need, or accompanying a project throughout many - if not all - its lifecycle phases, fostering reuse and minimizing ambiguity). From the beginnings of the Active Phasing Experiment, through VLT instrumentation, VLTI infrastructure, Telescope Control System for the E-ELT, until Wavefront Control for the E-ELT, we show how stepwise refinements of tools, processes and methods have provided tangible benefits to customary system engineering activities like requirement flow-down, design trade studies, interfaces definition, and validation, by means of a variety of approaches (like Model Checking, Simulation, Model Transformation) and methodologies (like OOSEM, State Analysis)

Keywords: MBSE, SysML, OOSEM, State Analysis

1. INTRODUCTION

There are five main problem areas, as identified by NASA's Jet Propulsion Laboratory (JPL) [1], common to many large complex system projects:

- Mission complexity is growing faster than our ability to manage it …increasing mission risk from inadequate specification & incomplete verification
- System design emerges from the pieces, not from an architecture …resulting in systems which are brittle, difficult to test, and complex and expensive to operate.
- Knowledge is lost at project lifecycle phase boundaries …increasing development cost and risk of late discovery of design problems.
- Knowledge and investment are lost between projects …increasing cost and risk; damping the potential for true product lines
- Technical and programmatic sides of projects are poorly coupled …hampering effective project decision-making; increasing development risk.

A Systems Engineer takes care of seven major tasks: State the problem, Investigate alternatives, Model the system, Integrate, Launch the system, Assess performance, and Re-evaluate. Those tasks are defined by the International Council On Systems Engineering (INCOSE[8]) which was established in the beginning of the 1990's in response to the ever increasing complexity of modern systems and the need to recognize systems engineering as a profession. Its mission is to advance the state of the art and practice of systems engineering in industry, academia, and government by promoting interdisciplinary, scalable approaches to produce technologically appropriate solutions that meet societal needs. As part of this mission several strategic initiatives have been started, where one is the Systems Engineering Vision defining a 15 years view of the evolution of the systems engineering discipline. This initiative addresses several areas where the relevant area for this paper is Model-Based Systems Engineering (MBSE). It is considered one of the main means to tackle problems of current and future systems development.

* gchiozzi@eso.org; phone +498932006543; eso.org

Modeling, Systems Engineering, and Project Management for Astronomy VI,
edited by George Z. Angeli, Philippe Dierickx, Proc. of SPIE Vol. 9150, 91500L
· doi: 10.1117/12.2055540

INCOSE defines Model-based systems engineering (MBSE) as the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases [19]. When information about the system to be developed is rigorously and formally captured, this information can be queried, reasoned about, validated, traded, and transformed into other form for further processing.

Rigorous system engineering can be achieved by defining:

- Ontologies and Conceptual modeling
- Languages, Standards and Tools
- Methods and Processes
- Model Transformations

Ontologies and Conceptual modeling provide a well defined vocabulary for the system engineers to capture information with a precise meaning. This creates consistency and correctness across modeling artifacts and engineering domains for analysis, design and the resulting documentation, for the integration of a multiplicity of system views and models, and for the reuse across projects.

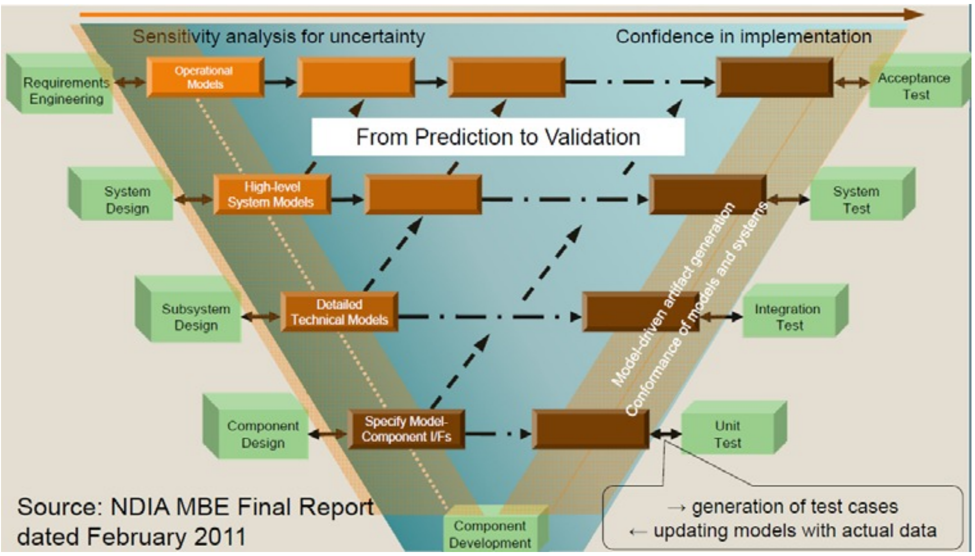
The use of industry standards (e.g. SysML) enables the embedding of this vocabulary into well defined graphical modeling languages and the use of the available tools.

Methods and process guide the modeler to develop a system model using languages like SysML and allows the engineer focusing on domain problems.

Model transformation is a key technology promoting the single source of truth paradigm and use the same information consistently for different purposes.

The problems of a traditional V-model development process are mitigated by model based practices [23]. The main challenge of the traditional process is that only the right leg of the V exposes problems incurred in the left leg due to validation. MBSE fosters the validation of requirements, architectural and detailed design already on the left leg early in the system development using system models (Figure 1).

One key enabler for MBSE is the Systems Modeling Language (SysML) that is a graphical modeling language designed to support System Level modeling and integrates Requirements, Structure, Behavior and Analysis [13].


Sensitivity analysis for uncertainty
Confidence in implementation
Requirements Engineering
Operational Models
Acceptance Test
From Prediction to Validation
System Design
High-level System Models
System Test
Subsystem Design
Detailed Technical Models
Integration Test
Model-driven artifact generation
Conformance of models and systems
Component Design
Specify Model-Component I/Fs
Unit Test
Component Development
→ generation of test cases
← updating models with actual data

Source: NDIA MBE Final Report dated February 2011

Figure 1 -Enhanced V-Model

Rigorous and formal modeling has evolved in the last decade, in particular since SysML emerged in 2007. ESO has a long-standing experience in different fields of system modeling which is laid out in the next section.

2. HISTORICAL PERSPECTIVE

For the development of the control system of the Auxiliary Telescopes (ATCS) (1998-2004), a Use Case driven, architecture-centric, iterative and incremental development process (based on the Unified Development Process [5]), substantially supported by models using the Unified Modeling Language (UML) [6], was adopted in order to solve or mitigate problems encountered in the previous projects. Those were primarily related to capture and tracing of requirements, tracing of interfaces, and ensuring documentation consistency[2]. A UML model captured the system's Use Cases, the architectural, structural, and behavioral specification of the control software, ensuring full traceability across those artifacts. To guarantee consistency between models and documentation some effort was spent in 1999 to implement a "one document" approach [2] where HTML and Word documents were produced using Telelogic DocExpress from Rational ROSE models. Unfortunately the transformation framework offered insufficient control over the generated artifacts and therefore this approach was used only in the ATCS project. Ten years later a plug-in for MagicDraw, the Model Based Document Generator [3][4], was developed in-house with ownership over the transformation allowing full compliance with ESO documentation templates. The Model Based Document Generator (MBDG) is a profile and a plug-in developed by ESO for the commercial modeling tool MagicDraw to be able to write documents as SysML models and to transform them into DocBook [17] XML files. Since documents and system models coexist within the same modeling environment, duplication of information is avoided and consistency is automatically maintained. The generated DocBook files can be converted into different document formats such as PDF.

The ATCS development was also the first successful attempt of applying model transformation (to generate production code) to the development of telescope control software using the Local control unit Server Framework (LSF) tool. LSF was created in 2000 to help building applications running on the real-time local control units accessing the hardware. This was the beginning of a model driven development process which has evolved substantially over the years and has become an important part of MBSE for control system development. In 2004, inspired by LSF, a tool suite called Workstation Software Framework (WSF) was developed to generate soft real time supervisory applications [7] from Rational ROSE and MagicDraw UML State Machine models. WSF was initially created to build the supervisory applications of the control software for the PRIMA VLTI facility and later successfully adopted for the development of applications for many other projects of the VLT program such as the Interferometric Supervisor Software configuration process, the Delay Lines rail-alignment tool, the APE project, and the New Generation CCD (NGC). In 2009, based on the experience gained with WSF, a new project was started to create a platform independent transformation tool to develop state machine driven applications.

Model transformation has become over the last years a key to modern control software development and also for systems in general. Model driven control system development is of greatest importance for the system as such because it specifies and implements a huge fraction of the system's behavior. In order to maximize the benefits we strive to create several related development artifacts (documents, code, test cases, simulators, etc.), using model transformation, from the same model and thus move gradually to a model based engineering approach. Model transformations (languages) play a very important role while model simulation is used for rapid prototyping and analysis of the dynamic behavior of groups of applications. Model checkers allow the formal verification of properties of the system and the design of test cases using the traces of the model checker. An important step has been to start generating documentation from the model which ensures consistent documents (no need to cross-check all the time deliverables such as ICDs, requirements, and design documents) and building up an infrastructure for model execution and code generation with the final goal to have an integrated system model which allows us to specify structure, behavior, generate code, and deployment configuration.

In INCOSE's MBSE framework ESO has collaborated with the German Chapter of INCOSE (GfSE) forming a "MBSE Challenge Team". The task was to demonstrate solutions to "challenging", non-trivial, real-world problems using MBSE. The SE^2 Telescope Modeling challenge team was founded in 2007 [10] with the following goals:

- Provide examples of SysML, common modeling problems and approaches.
- Build a comprehensive model, which serves as the basis for providing different views for different engineering aspects and associated activities.

The system case study was a technology demonstrator for the future Extremely Large Telescope (ELT), called the Active Phasing Experiment (APE), which is a high-tech interdisciplinary opto-mechatronical system. The activities included the creation of modeling guidelines and conventions for all system aspects, hierarchy levels, and views, and of the corresponding fully fledged SysML model, which was later regarded as an educational model because it shows the real system parts and the corresponding model elements. The main results, collected in a cookbook [10], were:

- APE system model, guidelines (domain-specific and general), and best practices. Examples included naming conventions, application of appropriate model annotations, proper use of modeling constructs, and model reuse considerations.
- Model libraries (e.g. for electronic equipment), and Profiles; Customization of the modeling tool
- Input for tool vendor and SysML Revision Task Force of the Object Management Group (OMG)

Figure 2 is an example from the SysML cookbook for MBSE [10] (based on SysML 1.3) of interface modeling in the telescope domain. The Service Connection Point (SCP) is an interface between the telescope and an instrument which provides to the instrument services as power, network and liquid cooling. On the left side you can see a photo of the SCP, while on the right you have its model, when looking inside the black box.

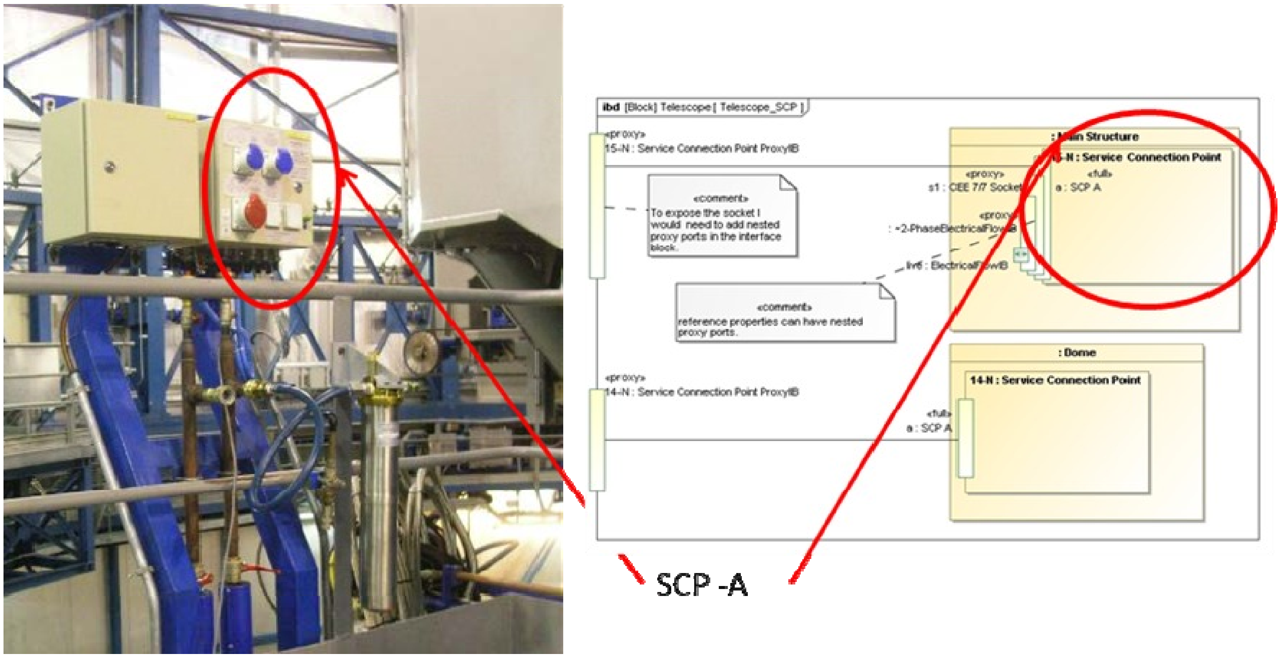

ibd [Block] Telescope [Telescope_SCP]
«proxy»
15-N : Service Connection Point ProxyIB
«comment»
To expose the socket I would need to add nested proxy ports in the interface block.
«comment»
reference properties can have nested proxy ports.
«proxy»
14-N : Service Connection Point ProxyIB
: Main Structure
15-N : Service Connection Point
«proxy»
s1 : CEE 7/7 Socket
«proxy»
: ~2-PhaseElectricalFlowIB
Inf6 : ElectricalFlowIB
«full»
a : SCP A
: Dome
14-N : Service Connection Point
«full»
a : SCP A
SCP -A

Figure 2 - Interface modeling

The Telescope Challenge team received an INCOSE working group award in 2010 for its contribution in achieving the systems engineering vision.

In the year 2008 we engaged in a two-year campaign of preliminary design for the European Extremely Large Telescope (E-ELT), with many tasks to be carried out, and a number of artifacts and deliverables to be produced for the Telescope Control System (TCS). The defined tasks (implicitly setting upfront the scope of what we wanted to achieve by system modeling) provided the ideal test bed for MBSE to prove it could enable us to deliver the required items in time and with high quality. . This made it possible to determine when modeling is complete.

For the construction proposal released at the end of 2010, several deliverables were based on information from a common project model (more details can be found in [4]).

The main principle of our modeling efforts was to strive for a pragmatic approach: we do not want modeling for the model's sake. One important decision to take was to define what we wanted to get as a result out of the modeling and therefore to prove termination criteria for the modeling activities.

However, this has turned out to be more difficult than thought. In 2007, the main problem was that there was basically no material on the application of SysML available, apart from the specification. Meanwhile there are several books and plenty of tutorials available, starting from[13].

When we started to adopt MBSE practices for the E-ELT in 2008 [9], one of the first things we did was to model something we knew well – it was the wave front control of the existing VLT telescopes. We learned what we wanted to model and how to model it, before we started modeling the E-ELT. When comparing Figure 3 and Figure 4 (the content of the figures is not relevant to the reader), it becomes evident that the essential complexity of the system has increased. It is the outcome of model queries regarding the number of interactions, activities or interfaces which can more objectively confirm this impression.

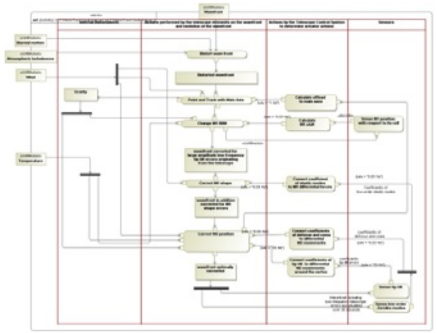


Figure 3 - VLT Wave front Control

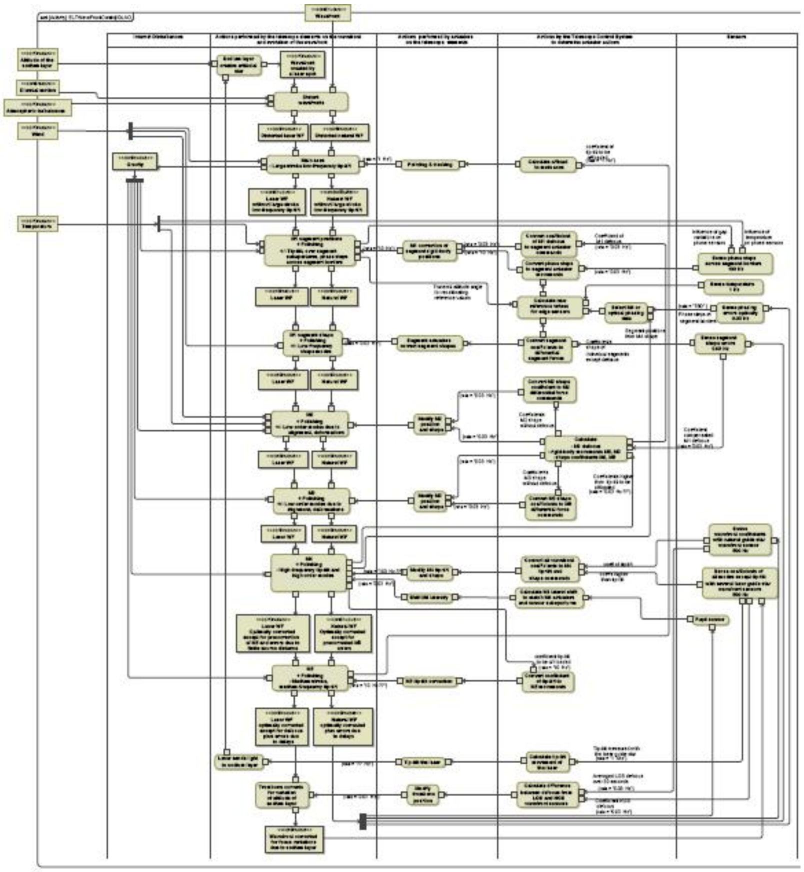


Figure 4 - ELT Wave Front Control

The modeling effort for the TCS made it very clear that a guiding methodology is required to efficiently apply MBSE.

In the beginning the system model focused almost only on the physical and behavioral aspects. The model was constructed according to guidelines [10] developed by the Telescope Modeling challenge team, which concentrated mostly on properly using SysML and organizing a large model without following a particular method. The resulting model served its purpose for the construction proposal, but it became manifest that a more systematic approach was required to support the development process in all its different phases, and in particular to support the design of the control for complex dynamic systems, covering both functional and physical models. We augmented the guidelines with two methods of proven worthiness: State Analysis (SA) [11][12] and the Object Oriented Systems Engineering Method (OOSEM)[13].

The status of the ELT-TCS MBSE effort can be summarized as follows. The control system team of the E-ELT delivered at the end of 2010 the construction proposal which consists of a complete preliminary design. The tasks for the successful completion of construction proposal included tasks like defining the control system’s infrastructure, power budgets and cost estimates from model based equipment catalogs. The results consisted, among other deliverables, of requirements and ICDs for contracted control systems, an architectural design of the TCS, and design conventions. The goals of enforcing systematic architecture rules correct-by-construction and a consistent verifiable system design have been being gradually achieved by adopting a formal system modeling language (SysML), and fostering the utilization of a common system model. A significant step was achieved by base-lining JPL's State Analysis Method and Architecture, and the continuous collaboration with JPL on a SysML profile for State Analysis[14]. The adoption of OOSEM concepts provides a more systematic approach to properly use the language, capture the required information, and organize it properly. The field testing at the VLT successfully started in 2010 and continued in 2011 by deploying and validating E-ELT technological choices, and applying State Analysis and OOSEM in an operational environment, refurbishing at the same time parts of the control system of the VLT. In 2012 and 2013 the TCS prototyping activities produced a prototype based on State Analysis architecture applying model driven development and code generation.

Some MBSE practices have been selectively adopted to address certain engineering problems in projects which are described in subsequent chapters. These concern the upgrade of the CRIRES instrument, the VLT 2nd generation fringe tracker, behavioral modeling of Wave Front Control Strategies, and the new generation Astronomical Site Monitor (ASM).

3. APPLICATIONS

This section provides some examples from the application of MBSE practices in different ESO projects.

3.1 VLT Upgrade

As part of validating the technological decisions for the E-ELT, control systems at the VLT are being refurbished using technologies expected to be used in the E-ELT control systems [18]. The upgrades serve several purposes: field testing technologies and methods in an operational environment (outside the lab), providing input into the E-ELT technology decisions, addressing obsolescence in the VLT telescope control systems, and preparing observatory technical staff for the construction of the E-ELT. As the first step in the upgrade and field test program, the Enclosure (Dome) Control System (ECS) of one VLT Unit Telescope (UT) was refurbished in 2010, using the State Analysis methodology and architecture. It is has been successfully operating at the Paranal observatory since then. The SA control pattern drove the development of the data model, which consisted of building a façade to the System Under Control, identifying and defining State Variables (SVs) and specifying Goals. Though only a limited and introductory application of SA was made, the architectural rules of SA aided in improving the design of the control system software. State Analysis is built on a sound theory which enables building an architecture for a distributed system, like the E-ELT, following well defined principles and rules. Although the scalability for very large systems still needs to be verified, the first practical experiences on the VLT give us confidence to pursue the path of applying SA for the VLT upgrade and eventually for the E-ELT control system.

3.2 Wave front control behavioral modeling

For the purpose of documenting an ELT wavefront control (WFC) scenario, which is currently under study, a state machine for one of the observing modes (Single Conjugate Adaptive Optics (SCAO)) was set up. The aim was to deliver a high level description of the WFC strategy that could be easily discussed between the involved stakeholders, being both understandable and not ambiguous. Our experience with this work has been that state charts and activity diagrams (to describe the behavior that are carried out in the individual states) was the most appropriate approach to achieve our objectives.

As an example, Figure 5 shows a part of the state machine modeling the sequence of states from standing still to the SCAO control. The transitions of some specific states are guarded according to specific conditions: for example, the Guiding state can be reached only if the guide stars have been already acquired.

Figure 6 shows instead an activity diagram for the telescope slewing. The slew is triggered by the operator selecting a specific target. Subsequently, TCS provides a target position to the actuators of the main structure, the M1 position actuators and the kinematic stages of the mirrors. The target positions are corrected using a pointing model. Slewing of the individual sub-systems is done asynchronously. The activity ends once all sub-systems have reached their positions.

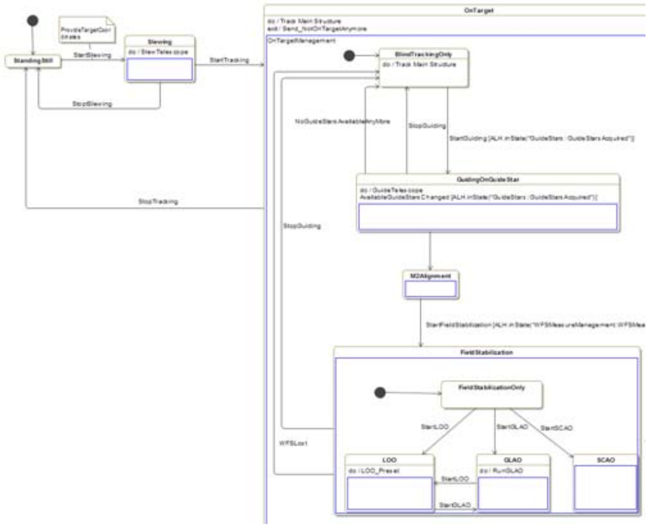


Figure 5 - Part of a potential state machine for the SCAO mode. Parallel regions are elided for readability

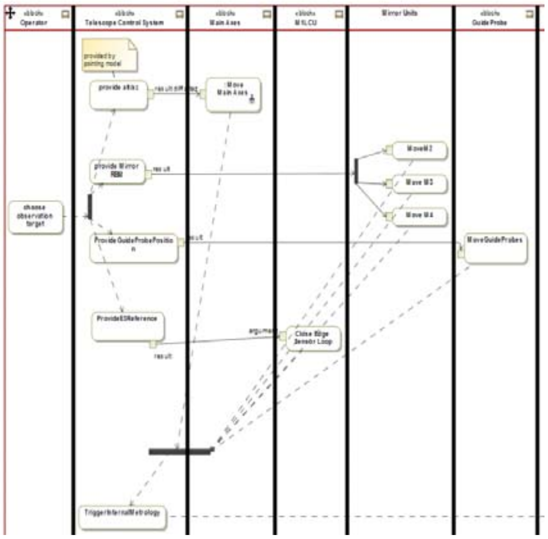


Figure 6 - Diagram for the telescope slewing activity.

3.3 CRIRES+, the CRIRES Upgrade

CRIRES, the CRyogenic Infra Red Echelle Spectrograph, is a popular VLT instrument offering a ground based high resolution (*R*=100,000) spectrographic capability in the 1-5μm wavelength range. However, its efficiency and effectiveness are hampered by the relatively small wavelength range covered by any single exposure. Hence an upgrade project is in progress. In addition a new polarimetric capability will be added as well as a number of other subsystem modifications.

Ideally we would have an existing MBSE description of the instrument from which we could identify the requirements, functions, interfaces, procedures etc. that need to be modified as part of the upgrade. Unfortunately we had no such description as a starting point, just a barely complete set of conventional written documentation. Consequently we set about implementing an *as-is* model (according to OOSEM's definition) of CRIRES using the MagicDraw tool. In the interest of efficiency, we limited this model to describing in detail only those parts of the system that shall be impacted by the upgrade. In particular we used the original technical specifications document to establish which specifications would remain unchanged (i.e. there was no need to model the associated functionality and interfaces) and which would be modified, and then added the entirely new specifications. This was a major undertaking, but immediately paid off in the sense that the very process of deciding how best to represent the existing instrument in MagicDraw lead to an improved understanding of the existing instrument (note that the team implementing the upgrade has very few members of the team who built the original instrument). This benefit arose largely because the MBSE approach lead to a more disciplined approach to describing the original instrument and lead us to seek out details of the as-built hardware and software that were not immediately clear from the conventional documentation.

Figure 7 gives an example of *as-is* and *to-be* models, in this case for the calibration selector subsystem, that have helped us to identify changes to the mechanical, optical, electronic and software interfaces required by the upgrade. This is very much a work in progress and we want to proceed in a pragmatic way; that is, issues that can be quickly dealt with via conventional analysis (i.e. non-MBSE) are dealt with that way. However, as the level of detail in the model is increasing, we are finding more ways that it can help our insight and decision processes.

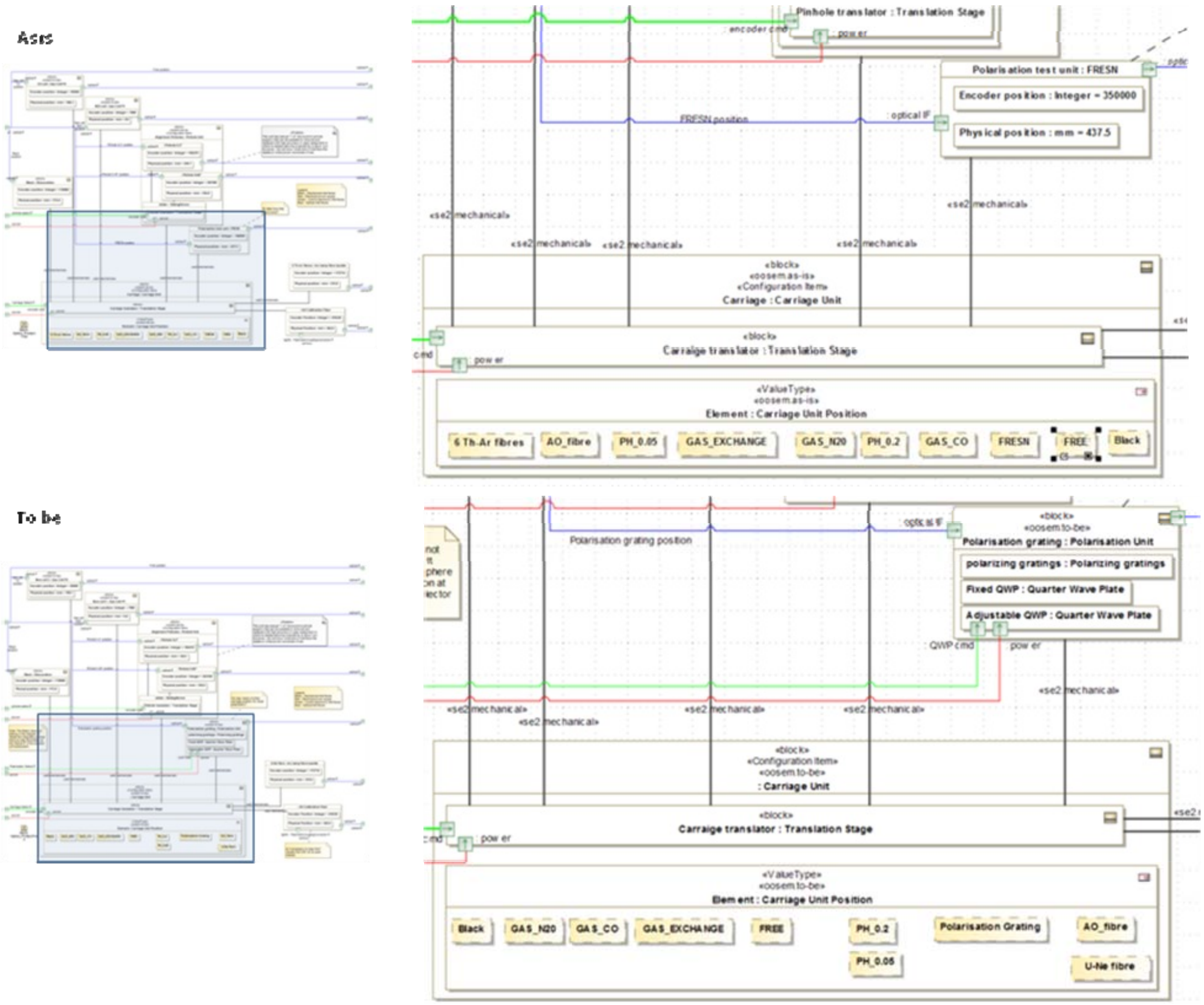


Figure 7 - CRIRES Selector Unit Subsystem as-is and to-be

3.4 VLT Interferometer

Since its first light in 2001 the Very Large Telescope Interferometer (VLTI) has arrived at a watershed transiting from the first generation of instruments into the second one. The instruments currently in operation will have to be de-commissioned soon in order to free laboratory space for the new ones coming. From a system engineering point of view this transition is quite challenging as it goes far beyond a pure decommissioning in the conventional sense. In addition to the removal of the old instruments the whole infrastructure has to be modified and refurbished to account for the needs and requirements of the new instruments. This involves a lot of changes to the overall observatory that – apart from the VLTI – shall be kept operational during this phase. While it is rather obvious that the interferometric instruments are complex systems, it could be easily overlooked that the VLTI itself can be considered as a complex system in its own with components such as telescopes, fringe trackers, delay lines etc.

Hence we are facing the challenges of integrating complex systems in a complex existing environment or system respectively. To master these challenges it appeared essential to identify the critical components that lie within the impact, problem solution and change area, to analyze the involved risks and to manage the various interfaces on the instrument as well as on the infrastructure side. Given this whole lot of different interdependencies we decided to employ a model based approach as we considered it superior for maintaining a consistent and linked point of view on the problem.

The model based approach was started within the scope of one particular VLTI project, namely the development of a second generation fringe tracker (2GFT). The aim was that this project would serve as a test-bed for MBSE at the VLTI; if successful, the modeling would be extended to other VLTI projects as well. In order to capture the operational domain of 2GFT we have started to model the VLTI *as-is* and *to-be*. Further, we added some use-cases as well as the various stakeholder requirements. The overall structure of the model was taken from ESO's OOSEM template. Unfortunately, at the point where we started to model the 2GFT system itself the project was temporarily put on hold due to shortage of manpower and changed priorities at ESO. We hope that we will be able to resume the modeling work in the near future. As far as we can say at this early stage, MBSE at the VLTI appears quite promising. However, we expect difficulties in interfacing the model based approach with the document based one still dominating ESO's systems engineering culture. Effective document generation from the model [3] will be key in this respect.

3.5 Astronomical Site Monitor

The Astronomical Site Monitor (ASM) Upgrade project aims at refurbishing the existing ASM in Paranal by replacing obsolescent components and adding new sensors in order to satisfy the requirements of the new generation of instruments coming to the telescopes at the observatory.

Devices more than 20 years old have to be replaced by up-to-date technologies. Some devices cannot be easily replaced or it is not meaningful to replace them. Moreover, the refurbished system has to appear unchanged to all existing telescopes and astronomical instruments: we cannot afford a cascade of changes on the current operational environment.

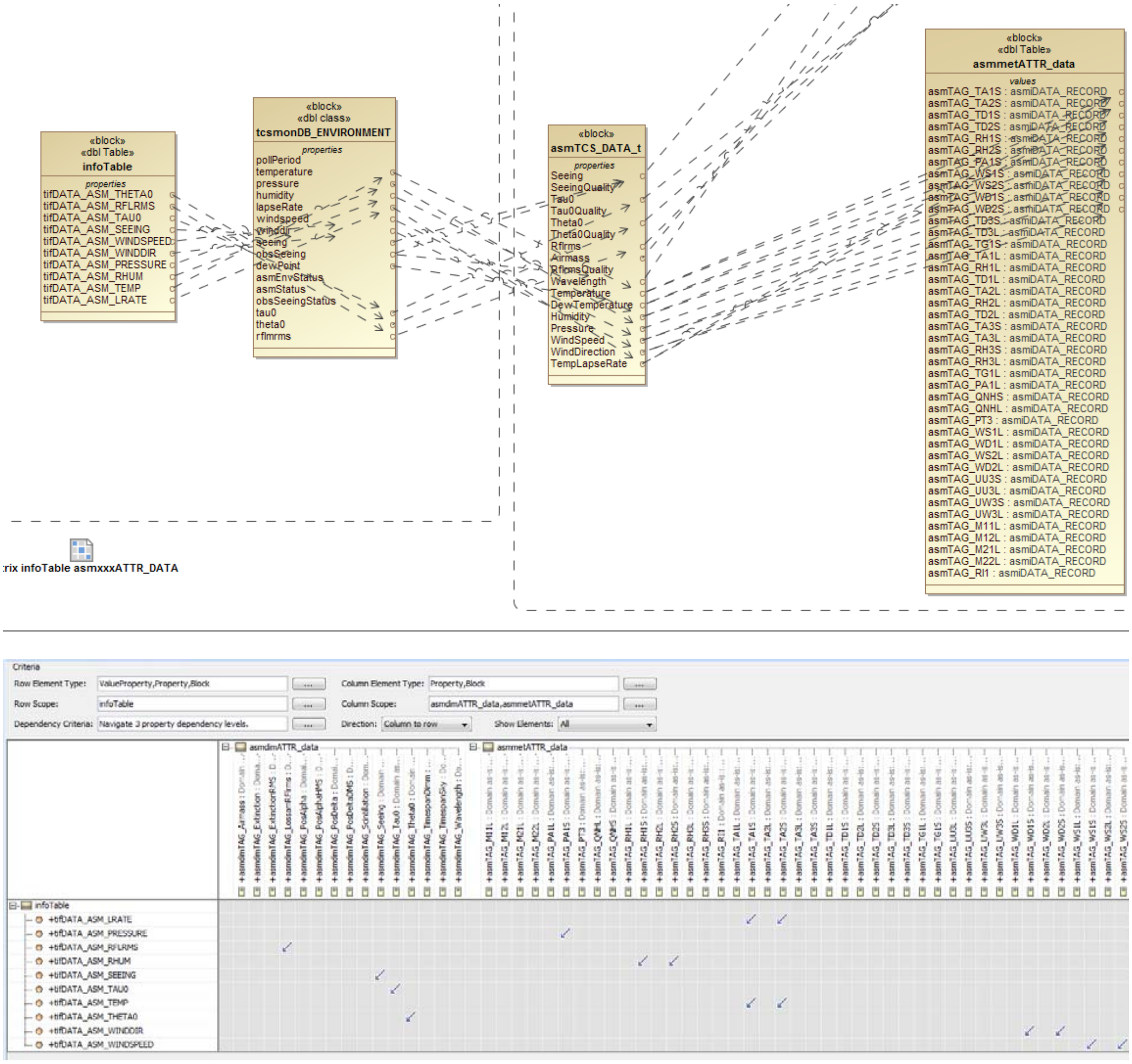
A major issue for us is to understand very well both external and internal interfaces among components of the *as-is* system, to evaluate the impact of the changes in the *to-be* system to avoid any interface backward incompatibility.

MBSE is helping us substantially in this task. Obviously we do not have an *as-is* model for the system designed 20 years ago, but there is no need to build one in order to solve our problems. What we are doing is to build slices of the models for the areas we need to analyze in detail, both for reverse-engineering the *as-is* system and to analyze the changes and to design the *to-be* system.

Just as an example, we have found extremely valuable the possibility of automatically identifying the complete flow of information from the ASM sensors to the existing instruments, using MagicDraw Dependency Matrices built using Meta Chain Navigation expressions [25]. Since the information produced by the ASM is going through several layers before reaching the astronomical instruments (like data collection and processing and the telescope control interfaces), it is not trivial to identify what is really used in the instruments and what could be the impact of changes. For us it has been sufficient to model the interfaces layer by layer and ask the MBSE tools to generate the full chain of dependencies for us.

In **Figure 8** you see on top the usage of data produced by sensors (right) flowing to instruments (left) through multiple interface layers. Below is the generated dependency matrix, showing the relations between the two ends of the flow.

We also use extensively the document generation infrastructure [3] to produce any document/memo for the project's stakeholders. We can at any time generate new versions of the documents if our analysis leads to changes in the model, so that our discussions with people not directly involved in the analysis and design can always take place on "traditional" paper documents, if this is for them more comfortable.


«block»
«dbl Table»
infoTable
properties
tifDATA_ASM_THETA0
tifDATA_ASM_RFLRMS
tifDATA_ASM_TAU0
tifDATA_ASM_SEEING
tifDATA_ASM_WINDSPEED
tifDATA_ASM_WINDDIR
tifDATA_ASM_PRESSURE
tifDATA_ASM_RHUM
tifDATA_ASM_TEMP
tifDATA_ASM_LRATE
«block»
«dbl class»
tcsmonDB_ENVIRONMENT
properties
pollPeriod
temperature
pressure
humidity
lapseRate
windspeed
winddir
seeing
obsSeeing
dewPoint
asmEnvStatus
asmStatus
obsSeeingStatus
tau0
theta0
rflmrms
«block»
asmTCS_DATA_t
properties
Seeing
SeeingQuality
Tau0
Tau0Quality
Theta0
Theta0Quality
Rfirms
Airmass
RfirmsQuality
Wavelength
Temperature
DewTemperature
Humidity
Pressure
WindSpeed
WindDirection
TempLapseRate
«block»
«dbl Table»
asmmetATTR_data
values
asmTAG_TA1S : asmDATA_RECORD
asmTAG_TA2S : asmDATA_RECORD
asmTAG_TD1S : asmDATA_RECORD
asmTAG_TD2S : asmDATA_RECORD
asmTAG_RH1S : asmDATA_RECORD
asmTAG_RH2S : asmDATA_RECORD
asmTAG_PA1S : asmDATA_RECORD
asmTAG_WS1S : asmDATA_RECORD
asmTAG_WS2S : asmDATA_RECORD
asmTAG_WD1S : asmDATA_RECORD
asmTAG_WD2S : asmDATA_RECORD
asmTAG_TD3S : asmDATA_RECORD
asmTAG_TD3L : asmDATA_RECORD
asmTAG_TG1S : asmDATA_RECORD
asmTAG_TA1L : asmDATA_RECORD
asmTAG_RH1L : asmDATA_RECORD
asmTAG_TD1L : asmDATA_RECORD
asmTAG_TA2L : asmDATA_RECORD
asmTAG_RH2L : asmDATA_RECORD
asmTAG_TD2L : asmDATA_RECORD
asmTAG_TA3S : asmDATA_RECORD
asmTAG_TA3L : asmDATA_RECORD
asmTAG_RH3S : asmDATA_RECORD
asmTAG_RH3L : asmDATA_RECORD
asmTAG_TG1L : asmDATA_RECORD
asmTAG_PA1L : asmDATA_RECORD
asmTAG_QNHS : asmDATA_RECORD
asmTAG_QNHL : asmDATA_RECORD
asmTAG_PT3 : asmDATA_RECORD
asmTAG_WS1L : asmDATA_RECORD
asmTAG_WD1L : asmDATA_RECORD
asmTAG_WS2L : asmDATA_RECORD
asmTAG_WD2L : asmDATA_RECORD
asmTAG_UU3S : asmDATA_RECORD
asmTAG_UU3L : asmDATA_RECORD
asmTAG_UW3S : asmDATA_RECORD
asmTAG_UW3L : asmDATA_RECORD
asmTAG_M11L : asmDATA_RECORD
asmTAG_M12L : asmDATA_RECORD
asmTAG_M21L : asmDATA_RECORD
asmTAG_M22L : asmDATA_RECORD
asmTAG_RI1 : asmDATA_RECORD
rix infoTable asmxxxATTR_DATA
asmdimATTR_data
asmmetATTR_data
infoTable
+tifDATA_ASM_LRATE
+tifDATA_ASM_PRESSURE
+tifDATA_ASM_RFLRMS
+tifDATA_ASM_RHUM
+tifDATA_ASM_SEEING
+tifDATA_ASM_TAU0
+tifDATA_ASM_TEMP
+tifDATA_ASM_THETA0
+tifDATA_ASM_WINDDIR
+tifDATA_ASM_WINDSPEED

Criteria

Row Element Type: ValueProperty,Property,Block

Row Scope: infoTable

Dependency Criteria: Navigate 3 property dependency levels.

Column Element Type: Property,Block

Column Scope: asmdimATTR_data,asmmetATTR_data

Direction: Column to row

Show Elements: All

Figure 8 - ASM interface dependency matrix

3.6 Variable Curvature Mirror

A Variable Curvature Mirror (VCM) is a mirror used to adjust the position of the telescope pupil by changing its radius of curvature using compressed air. VCMs are used in the VLTI Star Separators and Delay Lines and include air regulators which limit the maximum air pressure. However, recently, one of the VLTI VCM exploded due to high pressure. This accident triggered an internal project to verify how a more formal approach to analyze system properties could help to guarantee system integrity. In particular a Statecharts model of the new VCM control software for the Star Separator was created (Figure 9) and validated using a Java Pathfinder model checker[21][22]. The transformation from the Statecharts model to the Java code compliant with Java Pathfinder has been performed automatically using the COMODO tool[20]. System properties to be verified were expressed in terms of state invariants (and manually translated to Java using assertions); for example, the active control of the air pressure (represented by the VCM/PROTOCOL/ONLINE/CONTROLLING substate) requires a valid reading of the current pressure (represented by the VCM/MEASURING substate). The model checker will try out all possible sequences of Statecharts events and stop in the case the CONTROLLING substate becomes active while MEASURING is not. From the traces produced by the model checker it is possible then to run a simulation and understand where the flaw originates. Using this methodology, several wrong initialization and failure sequences (not otherwise detected using model simulation) have been corrected.

The model checking approach ensures that all possible sequences of events are tested, however this may require considerable amount of memory and time depending on the model size and complexity. In order to speed up the analysis, the input model can be simplified by selecting higher abstraction levels or optimized models[24]. It is also important to verify that the model execution semantic adopted by the model checker is consistent with the one implemented in the final production code and the one used by the simulation tool to make sure that system behavior is the same regardless of the tool.

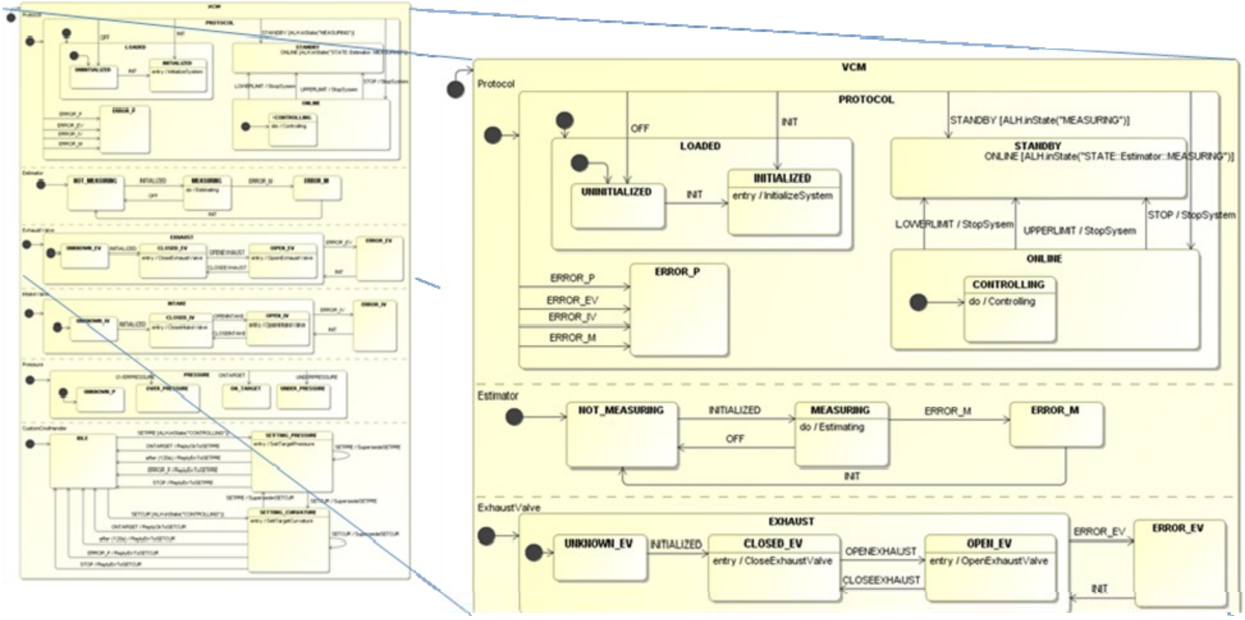


Figure 9 - Variable Curvature Mirror state chart

4. THE PILLARS OF MBSE

As mentioned before, the adoption of the foundational pillars of MBSE allows capturing information in a formal, consistent, and complete way addressing different aspects of the system in an integrated system model (Figure 10).

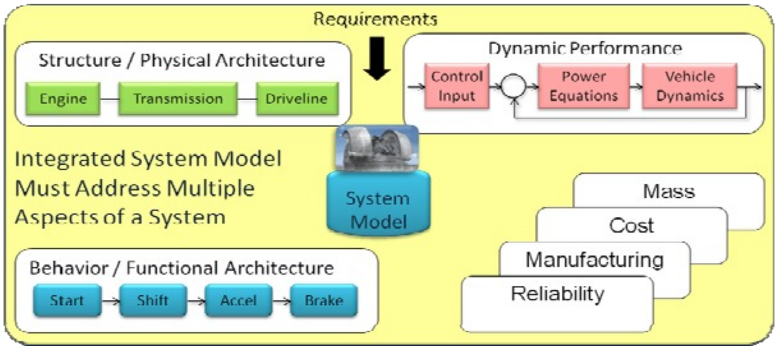


Figure 10 Integrating different aspects of a System

4.1 Ontologies and Conceptual modeling

Modeling a system from scratch may appear a daunting task at the beginning, since SysML (and the underlying UML) are complex, and too many choices are offered to modelers. Moreover, modelers come from different engineering disciplines.

On the other hand, as for any other large physics experiments organizations, ESO has developed well-established engineering *facts* or *business rules* about what a system should be like. For example, a scientific instrument must have a mechanical interface to the telescope hosting it.

These facts can be distilled by a system architect into engineering *business rules*, described in ontology formalism. The system architect uses this ontology to define the vocabulary of domain concepts whose words are drawn from the telescope engineering and control systems domains like *Instrument*, *Observing Block*, *Publisher*, *Control Unit* etc. A grammar of possible relations between such concepts, recurring across all our models, needs to be defined in the same ontology, e.g. *An Instrument is required to have a Relation to a Telescope through a Mechanical Interface*.

Some modeling tools (MagicDraw among them) offer immediate visual feedback to the modeler when modeling rules are being violated, this mechanism being extensible and configurable. The customized validation rules are directly pointing the modeler to a problem while creating the model.

It was therefore the next logical step to relate the vocabulary and the business rules to SysML stereotypes (contained in profiles) and Validation Rules (contained in tool plug-ins) respectively. This required a few preliminary steps.

In order to be able to formalize the business rules, a suitable UML Profile had to be introduced: we called this Conceptual Modeling Ontology (CMO), to act as a kind of Ontology Language, similar to OWL2 [15] to permit the expression of business specific concepts and relationships. It was developed by ESO based on work done by NASA/JPL [16] and ESO's own experience in defining DSLs using SysML. The elements it contains relate to foundational UML elements and are as generic as *Relation*, *Concept*, *Domain* (ownership), and *Range* (multiplicity).

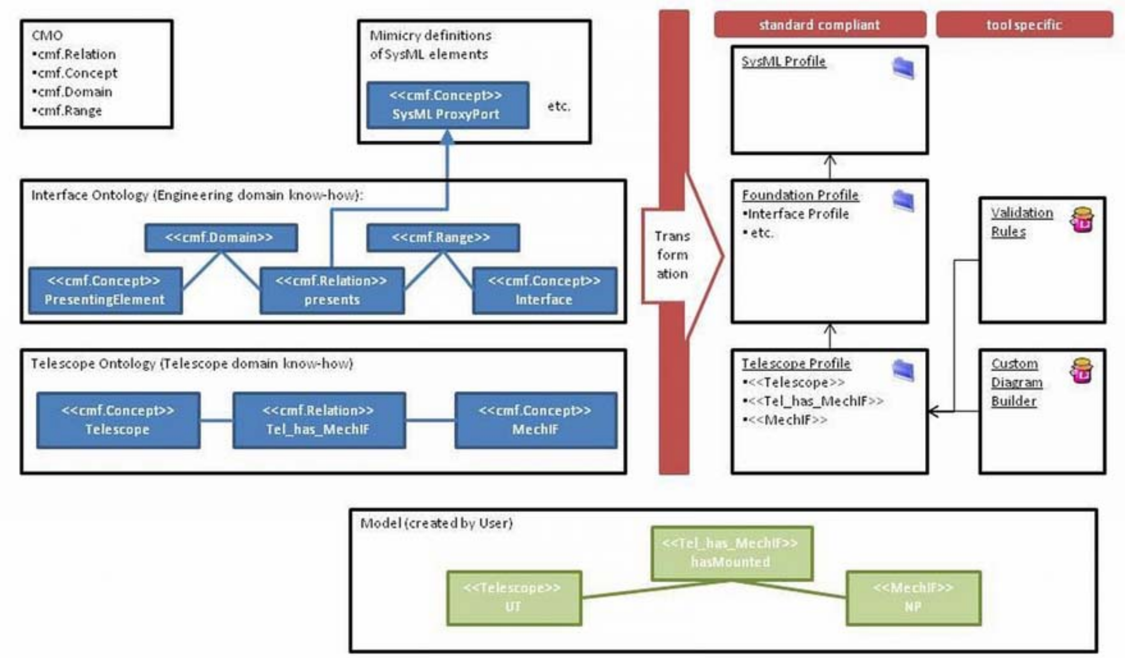
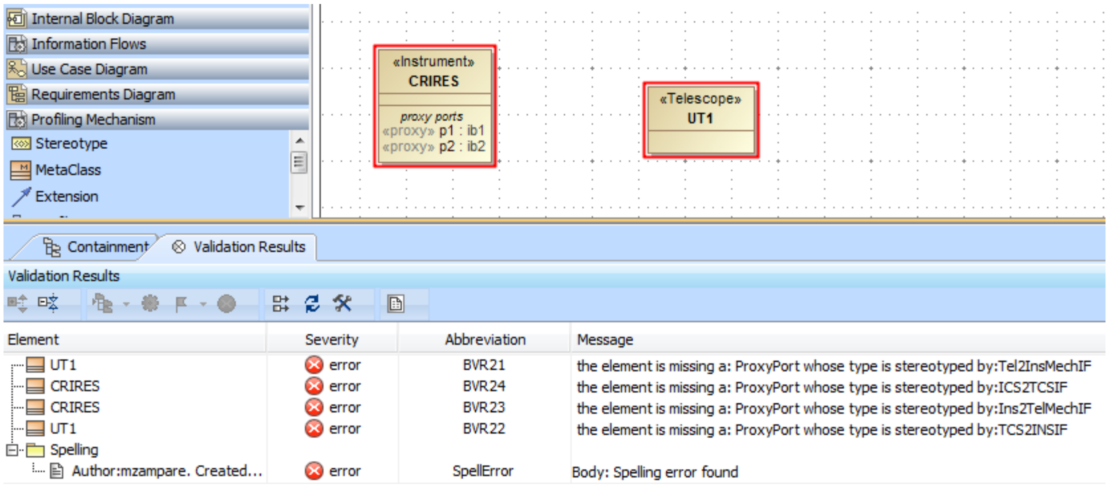


Figure 11 Interrelations between ontology, profiles and user model

Then, a transformation engine must was devised, which reads the ontologies and produces usable UML Profiles (tool independent) containing stereotypes for all the Concepts contained in them. The engine also produces validation rules such in a format such to be used as an extension of the specific validation suite existing in the modeling tool. The profiles are meant to be used in the end-user model, and the validation rules need to be installed as a plug-in to the modeling tool.

The algorithms of the transformation engine depend significantly on the CMO Profile, and various adaptations of our prototype turned out to be necessary when the latter had to be modified, as can be the case when the binding between the ontological building blocks (Concept, Relation, Domain and Range) and the corresponding UML elements (i.e. the meta classes for the corresponding stereotypes) was improved. With a stable mapping, the transformation engine merely needs to be re-executed when the business logic changes, in which case the end-user is required to update the model to the latest version of the generated profiles.

Our prototype transformation engine (dubbed Livingstone) accepts interdependent reusable layers of domain know-how, and transforms them into UML profiles and into custom validation rules for MagicDraw (Figure 11). Livingstone is beta, and not yet in production. Future work will involve, among others, the generation of custom diagram editors for MagicDraw. The user model is actively (immediate visual feedback) validated against the ontology and its rules which allows to create ontology compliant models by construction. Figure 12 shows an example of such a validation in MagicDraw where an element is missing for a proper definition of interfaces and therefore the concerned model elements are highlighted in red.



Element	Severity	Abbreviation	Message
UT1	error	BVR21	the element is missing a: ProxyPort whose type is stereotyped by:Tel2InsMechIF
CRIRES	error	BVR24	the element is missing a: ProxyPort whose type is stereotyped by:ICS2TCSIF
CRIRES	error	BVR23	the element is missing a: ProxyPort whose type is stereotyped by:Ins2TelMechIF
UT1	error	BVR22	the element is missing a: ProxyPort whose type is stereotyped by:TCS2INSIF
Spelling			
Author:mzampare. Created...	error	SpellError	Body: Spelling error found

Figure 12 Active Validation of User Model

4.2 Languages, Standards and Tools

The success of MBSE has a strong dependence on the capability to communicate with stakeholders and system implementers. SysML, a system modeling language standardized by OMG[27], provides the basic semantics to model interdisciplinary systems. Model based document generation puts the value of models into the deliverables systems engineers must produce, i.e. reports and documents. A modeling tool is required to adhere to the language standard, be customizable, use an open format for storage and support concurrent development and version control. The tools on the market expose a different level of standard adoption and definitely have a deficit when it comes to model interchange. But the reasons are more commercial than technical. The more the tool adheres to the standard the better it helps in applying properly SysML. With permissive tools (not enforcing sufficiently the SysML standards) the resulting model will most probably not be compliant with the language specification and therefore the added communication value will be heavily affected. In addition, the SysML standard is still evolving and exposes semantic ambiguities, gaps and lacks sometimes clarity that makes it more difficult for tool vendors and end-users to properly use the language.

4.3 Methods and Processes

The **State Analysis** (SA) methodology [11][12] is targeted to the control related domain, and focuses on behavior. It has been developed by NASA's Jet Propulsion Laboratory (JPL) since 1990. The methodology, which is founded on a state-based architecture and goal-based operation, defines a process for identifying and modeling the physical states (discrete and continuous) of the system under control and their relationships. State Analysis provides also a uniform, methodical, and rigorous approach for developing control system architecture. However, SA does not support SysML, and is not integrated into an overall system modeling process. Therefore, the first goal has been to develop a SysML profile [14] for SA in collaboration with the JPL.

The **Object Oriented Systems Engineering Method** (OOSEM) [13] integrates top-down functional decomposition with a model-based approach that uses SysML to support the specification, analysis, design, and verification of systems. OOSEM is intended to ease integration with object-oriented software development, hardware development, and test. It encourages use of OO models to capture system and component behavioral, performance, and physical characteristics that provide the basis for integrating other specific engineering models.

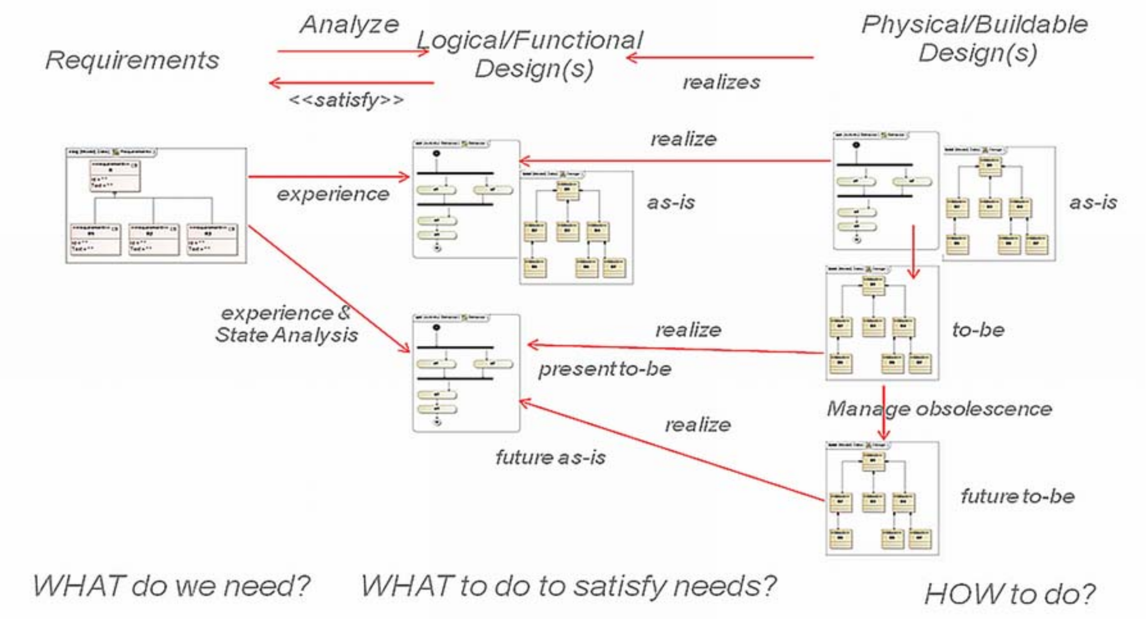


Figure 13 - Relation of Requirements, functional and physical Design

From both methodologies we apply the practices which we think provide most of the gain. The methods (in particular OOSEM) have been slightly adapted to our domain specific needs. The lesson learned is that a method shall be applied if and only if it provides structured guidance in the maze of MBSE. It is important to realize the necessity of different levels of abstraction, their relation and how to effectively use them.

The main benefit is seen in managing properly the evolution of systems at functional and physical level (Figure 13) with several generations of physical implementations and functional architectures according to Wymore[26].

4.4 Model Transformation

Using model transformation languages, (SysML) input models can be automatically transformed into artifacts (output models) to be used by other tools, such as simulators, analyzers, model checkers, (software) compilers, word processors, etc. In a model transformation, both input and output models are conforming to given meta-models. A meta-model specifies the abstract syntax and semantics of a set of models useful in a specific domain. We have seen that flexible and powerful model transformation tools (like Eclipse Xtend and Xpand) are now available and are key to exploit the benefits of an integrated system model.

5. CONCLUSIONS

It is very important to realize that there is more to modeling than just drawing diagrams. The model adds value to the system engineering practice because it can be validated, queried, reasoned about, and used to create documentation and operational artifacts like software. It is easy to fall into the trap of relying solely on diagrams. It is essential to have common and well defined semantic that is fully substantiated by the model behind the concrete syntax of the diagrams in order to create more artifacts from the same source in an automated way: this requires an expensive initial effort which pays back many times over later on. The application of appropriate MBSE methodologies for the problem domain (like SA and OOSEM) is valuable in defining common concepts and procedures, and keeping the resulting artifacts consistent in a common project model. Time must be set aside to build up an MBSE-friendly infrastructure; otherwise MBSE practices will also be considered a burden, because they are not well defined. Model transformation allows using different capabilities of different tools, starting from the same model, i.e. the same source of "truth".

REFERENCES

[1] Integrated Model-Centric Engineering (IMCE) Workshop for JEO, Jan. 2011.

[2] Chiozzi, G., Duhoux, P. Karban, R., "VLTI Auxiliary telescopes: a full Object Oriented approach", Proc. SPIE 2000, vol. 4009-03, p. 5, (2000)

[3] Model Based Document Generator, http://sourceforge.net/projects/mbse4md

[4] Karban, R., Zamparelli, M., Bauvier, B., Chiozzi, G., "Three years of MBSE for a large scientific programme: Report from the Trenches of Telescope Modelling", Proc. 22nd Annual INCOSE International Symposium, (2012)

[5] Jacobson, G.Booch, J.Rumbaugh, [The Unified Software Development Process], Addison Wesley, Reading MA,1998

[6] G.Booch, J.Rumbaugh, I.Jacobson, [The Unified Modeling Language User Guide], Addison Wesley, Reading MA,1998

[7] Andolfato, L., Karban, R. "Workstation Software Framework", Proc. of the Society of Photo-Optical Instrumentation Engineers, Vol. 7019, 70191X-1, (2008)

[8] International Council On Systems Engineering (INCOSE), http://www.incose.org

[9] Karban R., Zamparelli M., Bauvir B., Koehler B., Noethe L., Balestra A., 2008, "Exploring Model Based Engineering for Large Telescopes – Getting started with descriptive models", Proc. SPIE 7017, 70171I (2008)

[10] SE^2 Telescope Modeling Challenge Team, "Cookbook for MBSE with SysML", http://www.omgwiki.org/MBSE/doku.php?id=mbse:telescope

[11] Choi J., Coleman A., Dvorak D., Hutcherson J., Ingham M., Lee C.Y., Wolgast P., Feb 2008, "Goal-Based Operations of an Antenna Array for Deep Space Communication", iSAIRAS. Los Angeles, CA.

[12] Ingham M., Rasmussen R., Bennett M., Moncada A., Dec. 2005, "Engineering Complex Embedded Systems with State Analysis and the Mission Data System", AIAA Journal of Aerospace Computing, Information and Communication, Vol. 2, No. 12, pp-507-536

[13] Friedenthal S, Moore A., Steiner R., "A Practical Guide to SysML 2nd Ed", Morgan Kaufmann OMG Press, 2011

[14] Wagner D., Bennett M., Karban R., , "An Ontology For State Analysis: Formalizing the Mapping to SysML", IEEE Aerospace Conference, Big Sky, MT., USA, 2012

[15] OWL 2 Web Ontology Language, http://www.w3.org/TR/owl2-overview

[16] Jenkins, J., Rouquette, N., "Semantically Rigorous Systems Engineering Using SysML and OWL", 5th International Workshop on Systems & Concurrent Engineering for Space Applications, 2012

[17] Walsh, N.: "DocBook 5: The Definitive Guide", O'Reilly Media, April 2010

[18] Karban, R., Kornweibel, N., Dvorak, D., Ingham, M., Wagner, D., "Towards a State Based Control Architecture for Large Telescopes: Laying a Foundation at the VLT", 13th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPS), Grenoble, France, 2011

[19] Systems Engineering Vision 2020, INCOSE-TP-2004-004-02, Version 2.03, September 2007, CRC Press 1993

[20] Andolfato, L., Chiozzi, G., Migliorini, N., Morales, C., "A platform independent framework for statecharts code generation", 13th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPS), Grenoble, France, 2011

[21] M. Ben-Ari, "A Primer on Model Checking", ACM Inroads (March 2010, Vol. 1, No. 1)

[22] Mehlitz, P., "Trust Your Model - Verifying Aerospace System Models with Java Pathfinder". Proc. IEEE Aerospace Conf. '08, Big Sky, MT, Mar. 1-8, (2008)].

[23] NDIA, Final Report of the Model Based Engineering (MBE) Subcommittee (February 2011)

[24] Gibson, C., Karban, R., Andolfato, L., Day, J., "Formal Validation of Fault Management Design Solutions", Java Pathfinder Workshop 2013, Palo Alto, CA, USA (November 2013)

[25] Pavalkis, S., Nemuraite L., Butkiene R., "Derived Properties: A User Friendly Approach To Improving Model Traceability", Information Technology and Control, vol. 42, n.1 (http://dx.doi.org/10.5755/j01.itc.42.1.2470)

[26] Wymore, W., "Model Based Systems Engineering", CRC Press, 1993

[27] Object Management Group (OMG), http://www.omg.org