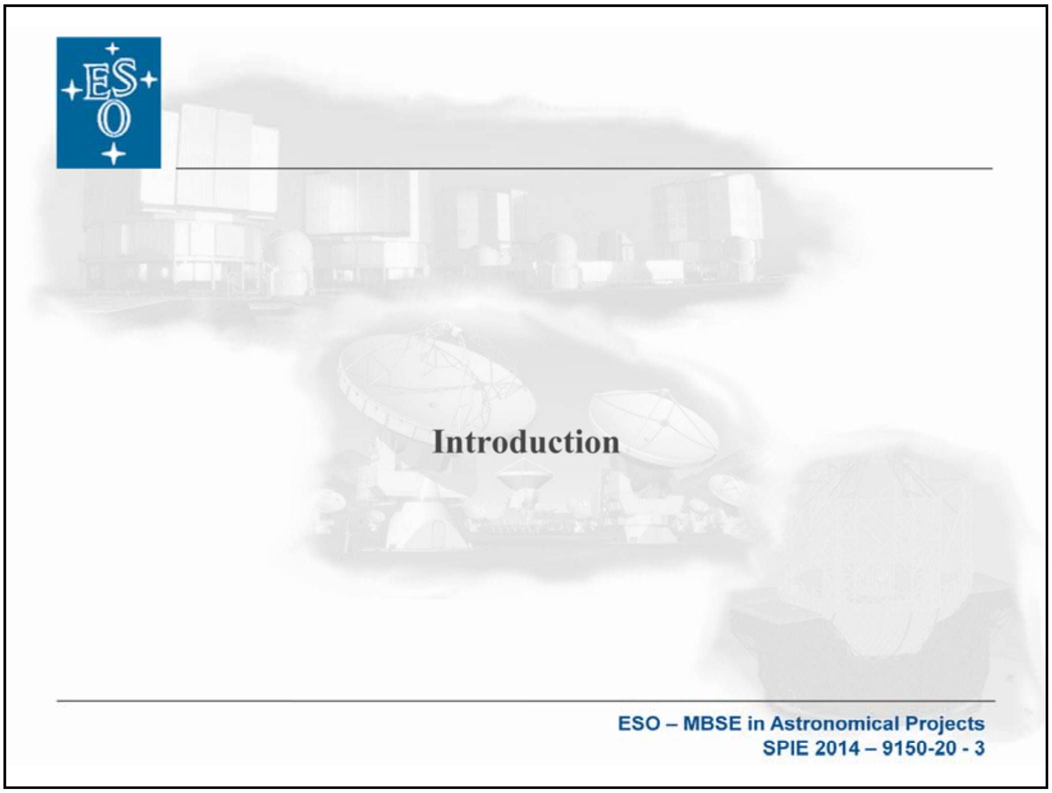
In this presentation I will surveys some of the capabilities, expectations and peculiarities of tools-assisted Model Based System Engineering, experienced in real-life astronomical project at ESO (starting from the first attempts to use a model centric approach to software development in 1998).

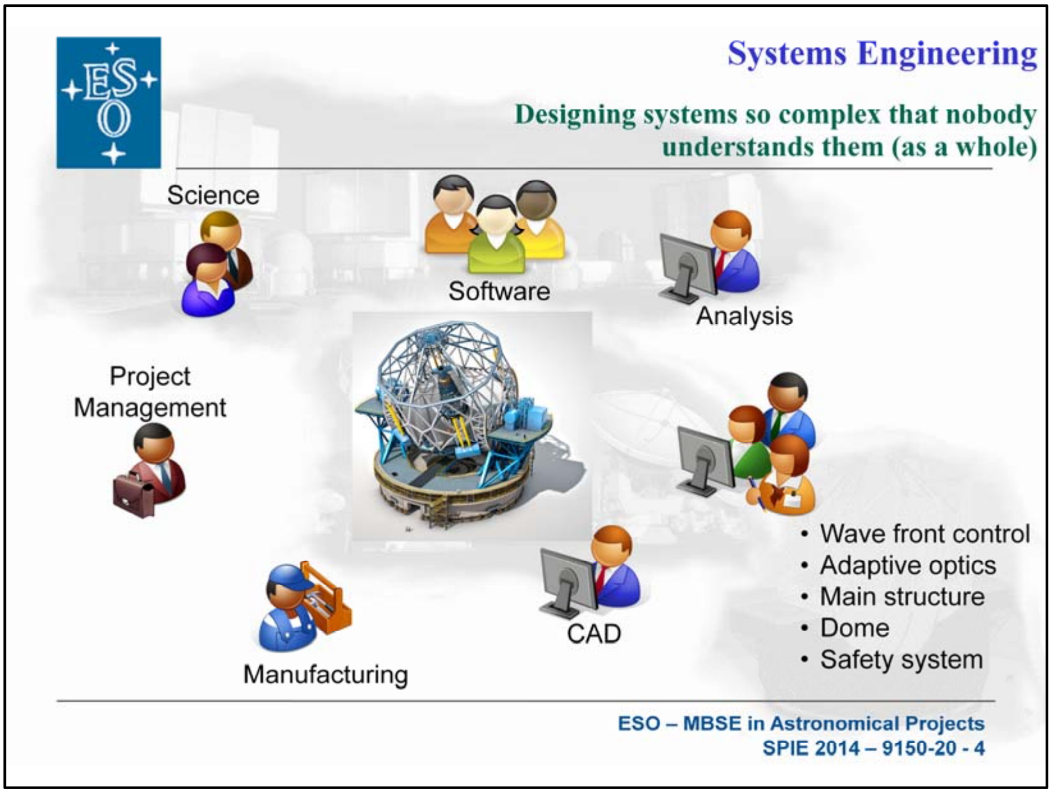
The main driver of this work has been R.Karban, who has put great dedication in mentoring and evangelizing MBSE in the organization for several years, allowing us to develop an internal culture in MBSE.

Unfortunately for us, he left ESO few months ago to take up MBSE/System Engineering responsibilities at JPL in Pasadena, but we understand that this is for him a great opportunity and we hope he will come back after some years with an even greater experience.

Outline

- Introduction
- The historical perspective
- Examples of applications
- The pillars of MBSE
- Conclusions


Introduction
ESO – MBSE in Astronomical Projects
SPIE 2014 – 9150-20 - 3



System Engineering as a discipline stems from the awareness that we are designing systems so complex that no single expert can understand them as a whole.

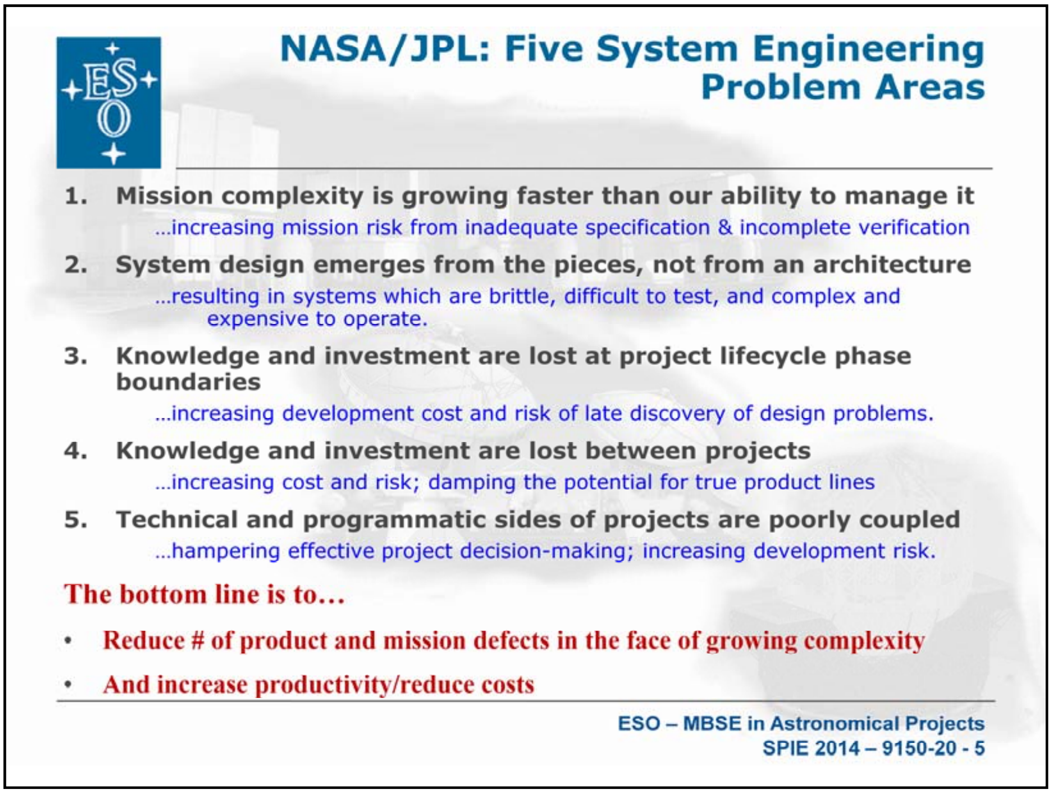
Many stakeholders are involved in the design and construction of such systems, all with their own view, interests, concerns and knowledge of the system.

We need therefore a robust, multidisciplinary, approach to analyze and elicit stakeholders' needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem, the system lifecycle.

Systems engineering is an interdisciplinary field of engineering that aims at formalizing such an approach. Systems engineering deals with work-processes, optimization methods, and risk management tools. It overlaps technical and human-centered disciplines such as control engineering, industrial engineering, and project management.

Systems Engineering aims at ensuring that all likely aspects of a project or system are considered, and integrated into a whole.

The System Engineering (team) in a project has to build the bridges across the different engineering and managerial disciplines, enabling communication, mutual understanding and coherence of the different perspectives.

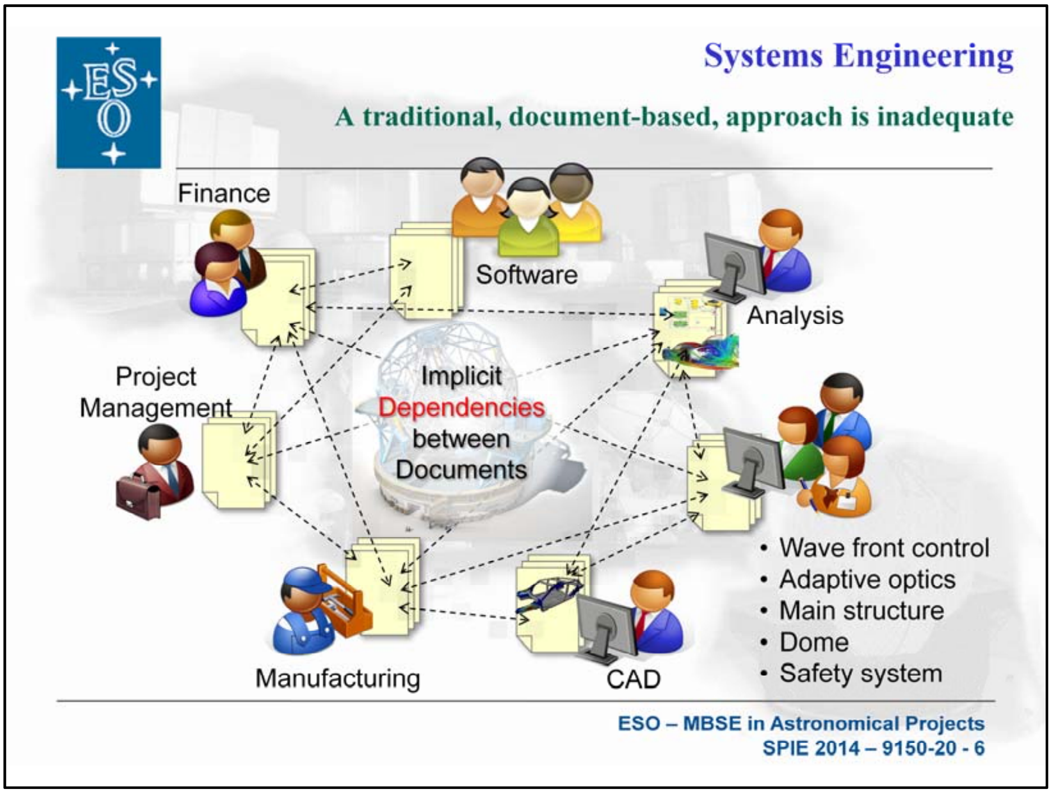


Going a bit more in details, what are the main problem areas that System Engineering has to tackle?

NASA/JPL has identified 5 main problem areas common to many large complex system projects.

……

How can System Engineering deal with these problems?



In the traditional, document-based approach, each stakeholder (engineer of a specific domain, project managers, scientists, procurement…) analyzes the system from his own point of view producing corresponding documents.

Often, domain specific models are produced for the needs of a specific viewpoint (CAD, FEM analysis, electronic diagrams), but they are used only for the specific domain work and eventually end up just as diagrams in documents.

But the reality is that they are all views of the SAME system → the views depend on each other.

With this approach, dependencies are often implicitly represented, hidden in documents, making inconsistencies very likely:

• Concepts are expressed with different terminology and become ambiguous

• Numbers might be different in different documents or have a different semantics

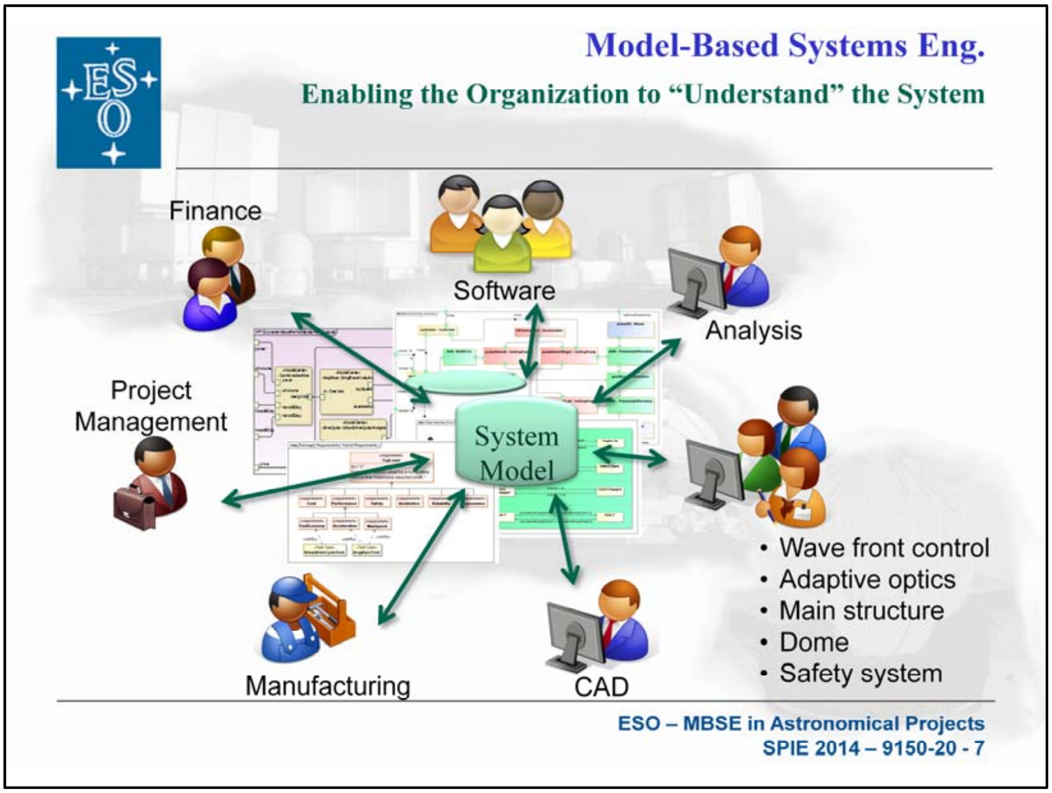
• Requirements might be in conflict

Can a single person build and keep up to date while the project evolves a coherent overview?

An engineer that cares for the project often spends a lot of time trying to understand what other engineers had in mind when they wrote the documentation for their aspects of interest.

Tools like DOORS have been developed to try to bring discipline in the process, but cover only some aspects, like requirements management.

Also, documents become soon obsolete and are not kept up to date once their immediate need in a specific phase of the project is over.



An alternative approach consists in creating and maintaining a **federation of models** which are tied together by a system model which keeps the abstract information of the system at an architectural level.

See for example SLIM for model integration (http://www.intercax.com/products/slim/)

Eventually the information in the SysML model becomes the information in the PLM (Product Lifecycle Management), tying all the models together.

SysML models can provide much richer information, including various relations.

- Stakeholders input their knowledge of the system in the model(s)

- Artifacts needed in the different phases of the project are produced from the model(s)

A formal model makes the dependencies explicit and makes the shared information more precise, less ambiguous and organized.

As we will see in more details in the rest of the presentation, an approach based on a central model has potentially a lot more advantages and possible usages. For example it might:

- allow to run analysis and tradeoffs

- allow to implement automated model checking

- provide an infrastructure for early testing

- allow to trace dependencies

But in practice this can work only if the effort to build and maintain the model is perceived by the stakeholders as considerably smaller than the benefits. More over, they must be able to extract from the model the information they need in a format that they understand easily.

For several years the processes and the technology have not been up to this task, but now things have changed:

- Stakeholders realize more and more that cannot cope with the growing complexity

- Tools are improving fast

- System Engineering processes are being refined.

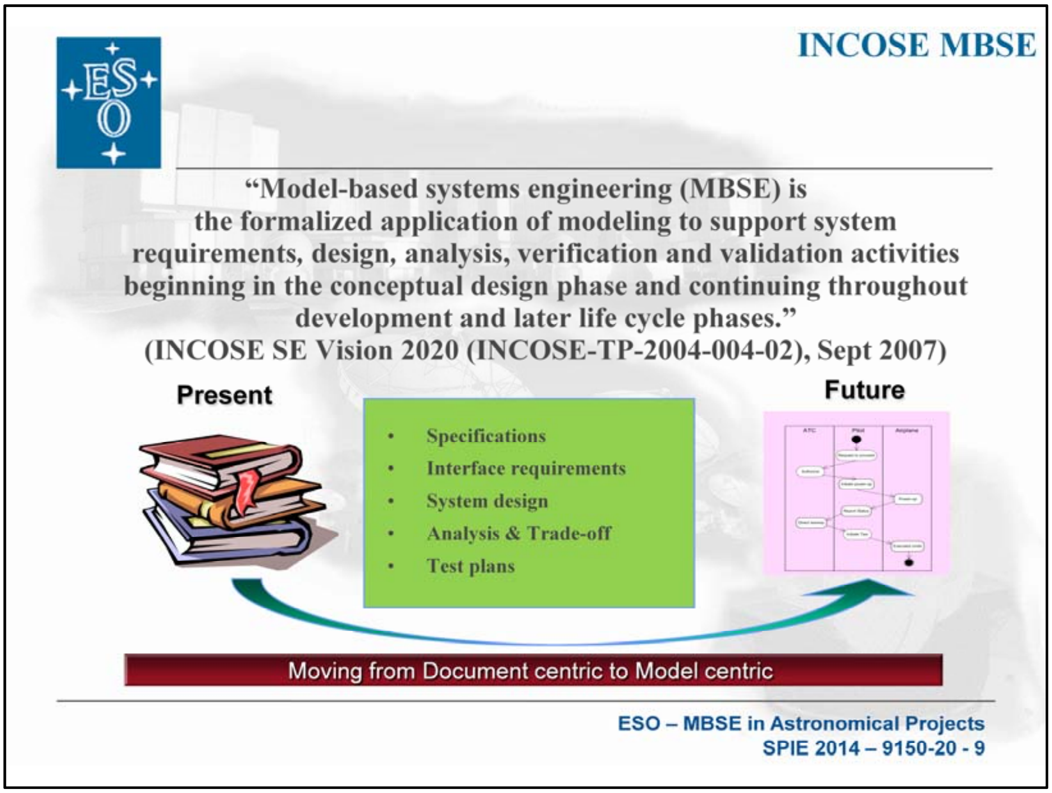
ICOSE has recognized the importance of this approach and has put in 2007 MBSE in its “SE Vision 2000” as one of the 5 focus areas for the status and future development of System Engineering, giving life to the MBSE Initiative.

The MBSE Initiative is organized into MBSE Challenge Teams and MBSE Activity Teams.

Challenge Teams are focused on demonstrating the application of MBSE to solve a particular problem that is identified by the team.

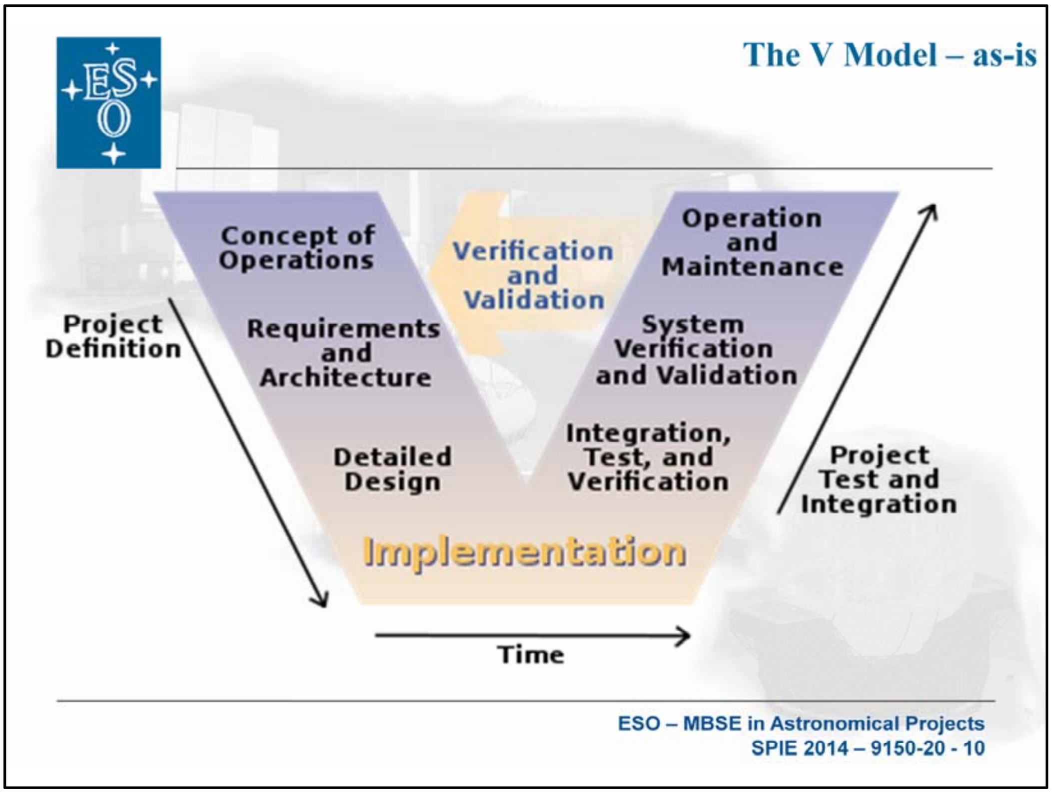
Activity Teams are focused on advancing a particular aspect of MBSE.

In this context, Robert has been leading the telescope modeling challenge team and collaborated closely with
the space systems team. Both received the INCOSE award together.



The INCOSE definition of MBSE clearly identifies the switch to a model centric approach.

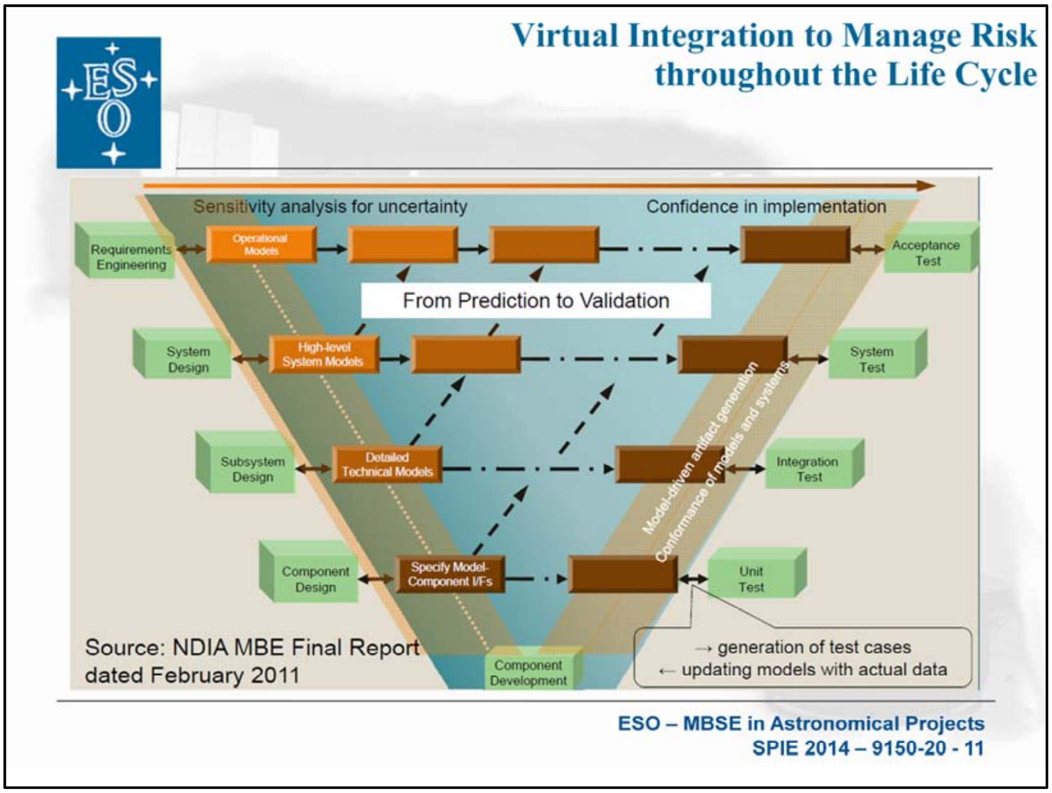
It also highlights the fact that the application of modeling shall be "formal", i.e. has to follow precise specified rules.



Model-based engineering tools and practices enable an adaption of the classic development “V.”

In the classical “V”, we simply decompose the system into its component down the left side of the V and then integrate and test these components up the right.

Integration and testing can take place only at a very late stage in the project.



MBSE enables incremental testing of the components during development, well before designs have been committed to hardware.

For example, it is possible to trace if requirements are addressed in the design, calculate budgets, check compliance of interfaces, run state machine simulations to find problems in the logic….

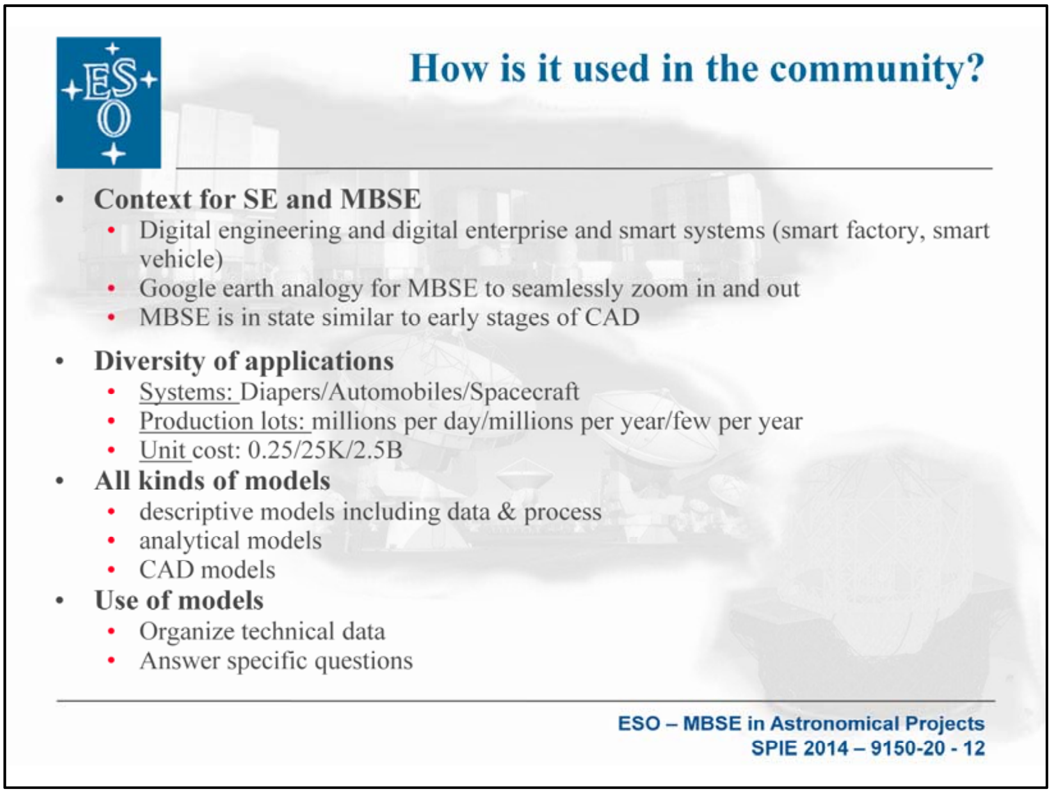
This introduces the ability to manage risk incrementally.

At each step in the refinement of the detailed design, systems, subsystem, and component requirements can be virtually tested against the current understanding of the design requirements and assumptions.

Tested and validated subsystem elements increase the confidence that the system will operate as intended, while tested and validated components elements increase the confidence that the subsystem will operate as intended.

Thus testing and validation can be more tightly coupled to the design activities allowing more rapid design convergence and validation of derived requirements in the same phase of the development lifecycle.

To improve the process further, there is the possibility of performing system integration tasks virtually as early in the development lifecycle as possible. This addresses the need to not only have components behave according to their specifications, but also that all the various components will integrate and lead to a system that behaves as intended.



Digital engineering and digital enterprise and smart systems (smart factory, smart vehicle)

Everything digitized, everything connected (internet of things)

Google earth analogy for MBSE to seamlessly zoom in and out

Integrate vertically and horizontally

Leverage IT technologies (visualization, connectivity, processing, ..)

Integrate discipline specific models

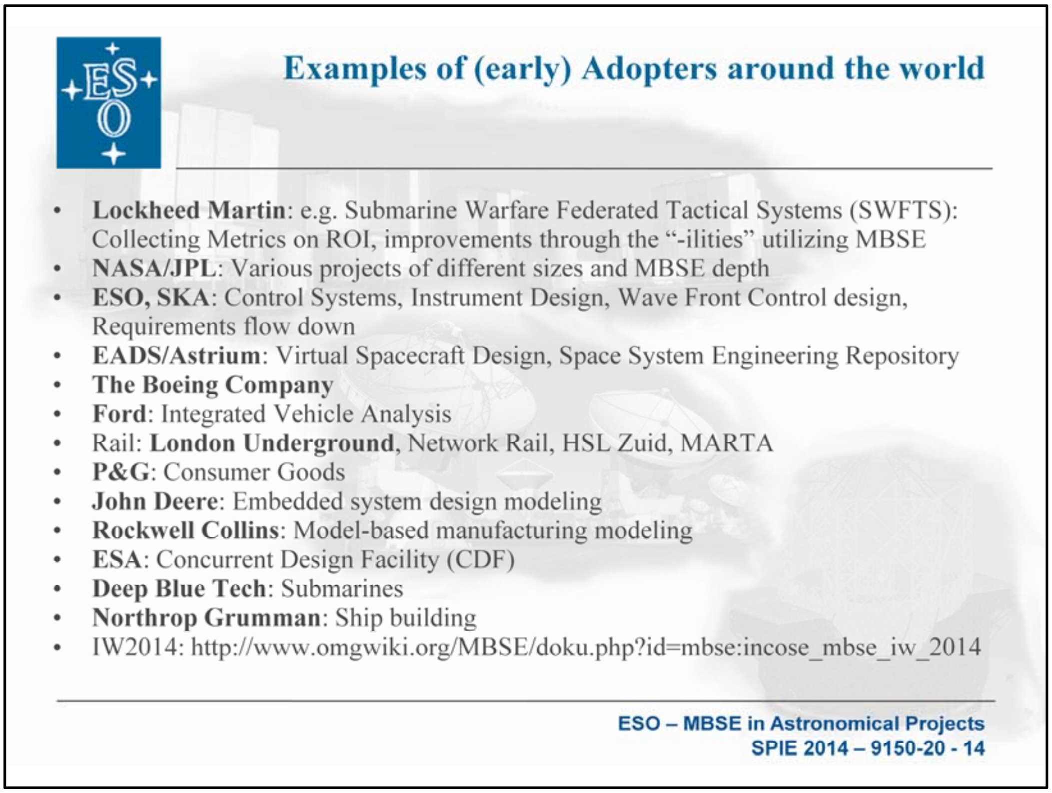
(e.g., integrated vehicle analysis)

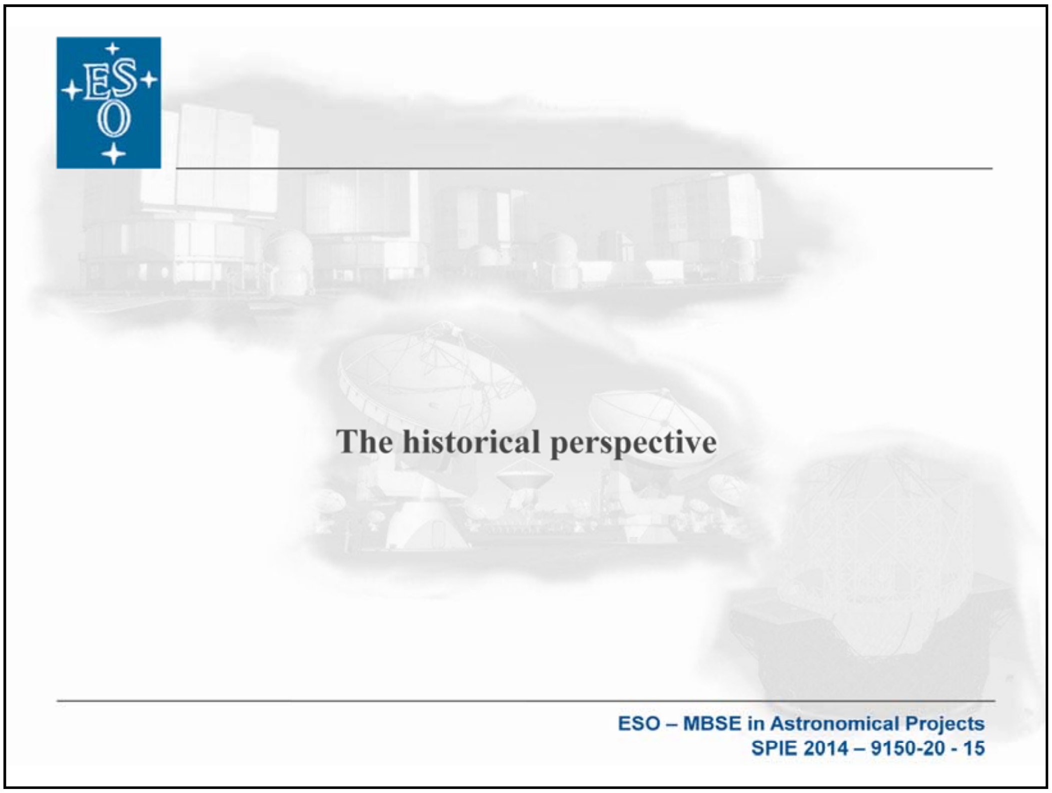
Communicate system information among stakeholders

domain engrs and disciplines (subsystems, ilities, mfg, …)

How is it used in the community?

- **System model**
  - Describe system
  - Integrate discipline specific models
- **System Model Benefits**
  - Communicate system information among stakeholders
  - Reuse of modeling information
  - Finding req'ts & design gaps/inconsistencies
  - Increased trade space exploration
  - Improved testing
- **Importance of MBSE methodology (e.g., systematic approach)**
  - Reqt's, functions, interfaces, features, components, analysis, verification
  - Analysis of behavior, performance, other attributes, failures
  - Use of models across life cycle
- **Challenges**
  - Tool integration
  - Visualization of models

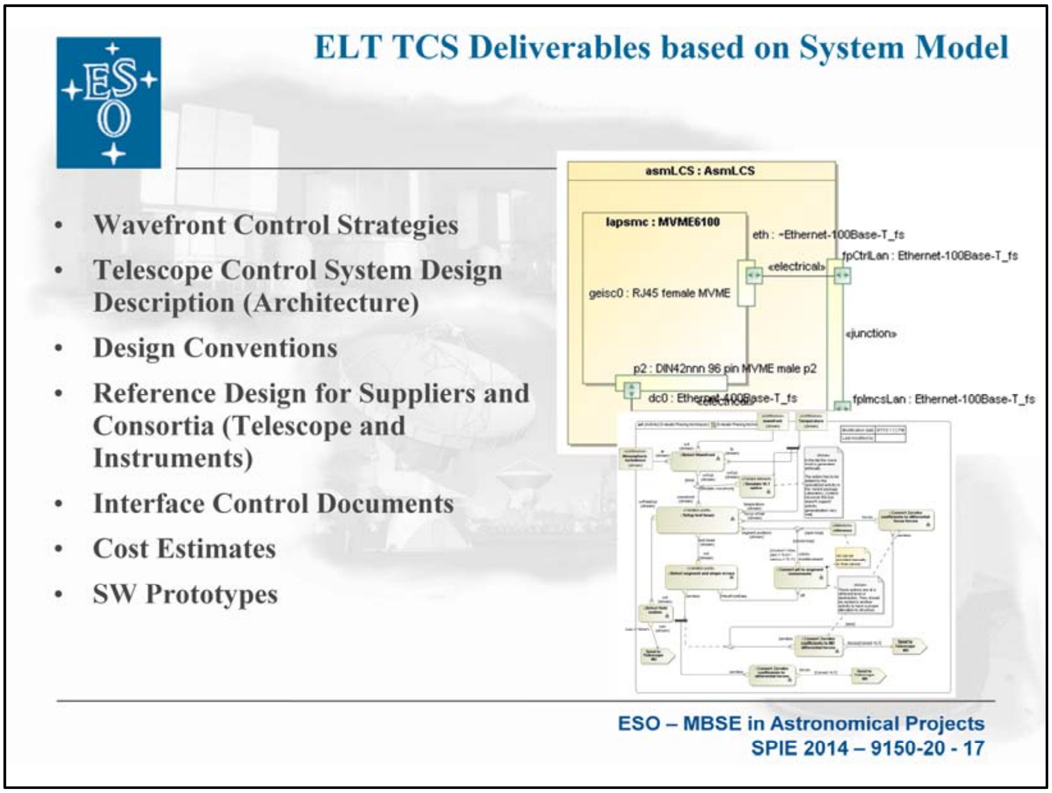

Examples of (early) Adopters around the world
• Lockheed Martin: e.g. Submarine Warfare Federated Tactical Systems (SWFTS): Collecting Metrics on ROI, improvements through the "-ilities" utilizing MBSE
• NASA/JPL: Various projects of different sizes and MBSE depth
• ESO, SKA: Control Systems, Instrument Design, Wave Front Control design, Requirements flow down
• EADS/Astrium: Virtual Spacecraft Design, Space System Engineering Repository
• The Boeing Company
• Ford: Integrated Vehicle Analysis
• Rail: London Underground, Network Rail, HSL Zuid, MARTA
• P&G: Consumer Goods
• John Deere: Embedded system design modeling
• Rockwell Collins: Model-based manufacturing modeling
• ESA: Concurrent Design Facility (CDF)
• Deep Blue Tech: Submarines
• Northrop Grumman: Ship building
• IW2014: http://www.omgwiki.org/MBSE/doku.php?id=mbse:incose_mbse_iw_2014
ESO – MBSE in Astronomical Projects
SPIE 2014 – 9150-20 - 14


The historical perspective
ESO – MBSE in Astronomical Projects
SPIE 2014 – 9150-20 - 15

1998 to present

- 1998 - Use Case Driven and Document generation for ATCS
- 2000 - LSF: Model transformation for ATCS
- 2004 - WSF: Model transformation for VLTI, APE, NGC, DLs
- 2007 - INCOSE's Telescope Modeling Challenge Team, APE
- 2008 - Construction Proposal for the ELT Control System
- 2012 - Prototyping for the ELT Control System

ESO – MBSE in Astronomical Projects
SPIE 2014 – 9150-20 - 16



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 extracted from a common project model.

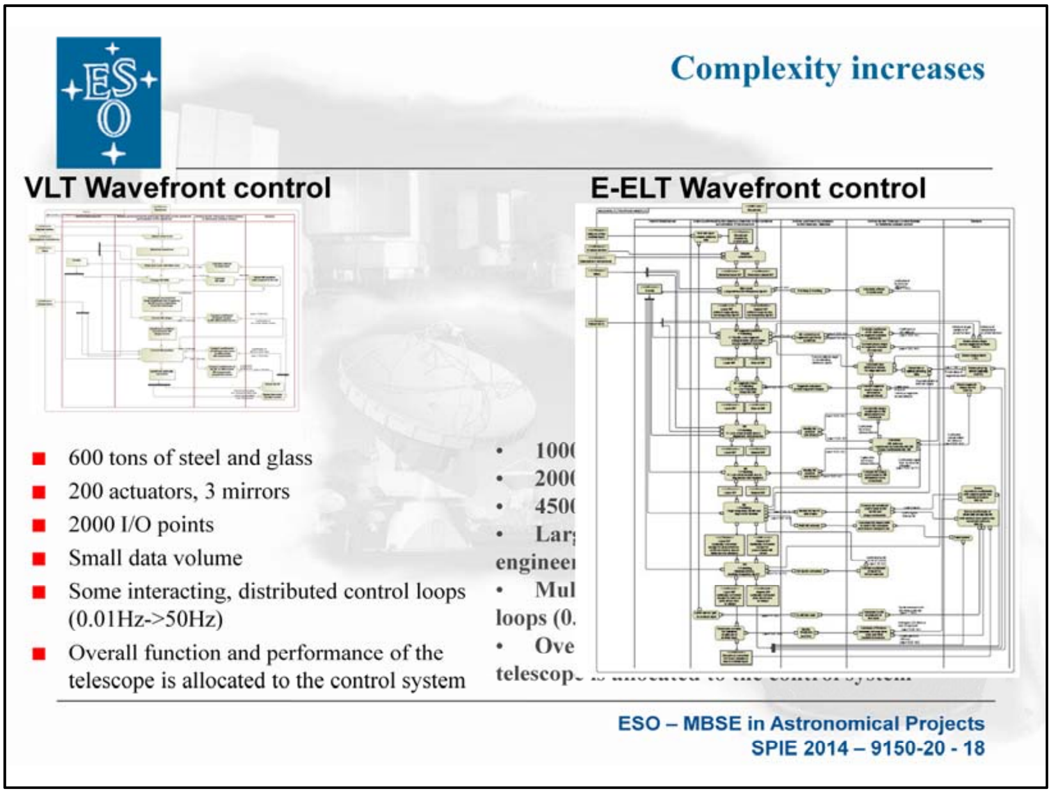
The main principle of our modeling efforts was to strive for a pragmatic approach: we do not want modeling for the model's sake. One important decision to take was to define what we wanted to get as a result out of the modeling and therefore to prove termination criteria for the modeling activities.

Some considerations can be done on these deliverables.

I find particularly interesting to briefly discuss ICDs.

In a typical document based system, you have to decide where the ICD is, in particular when it specifies the interfaces of parts contracted out: a separate document? A document on one of the two sides? Two separate documents, to take into account the different concerns of contractor and user without impacting contractual issues? This introduces a high risk of having inconsistencies on the two sides of the interface.

MBSE allows to specify the ICD with all details in a central place, producing specific views to address the concerns of the different stakeholders.



When we started to adopt MBSE practices for the E-ELT in 2008, 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.

We immediately saw that the evolution of Telescopes creates new challenges

• The VLT

  • Fact sheet (1st click)

  • WFC (2nd click): dominated by fixed client server relationships, basically 2 distributed control loops at moderate frequencies, overseeable number of components and system states

• The E-ELT

  • Fact Sheet (3rd click)

  • ONE of several WFC modes (4th click): tens of distributed control loops with hundreds of concurrent connections. Requirement to change at any time how the data is distributed and processed, adding/removing sensors at any time.

-> Many loosely coupled interacting components which cannot be forced into a traditional hierarchy with multiple layers.

In principle, every component can communicate with any other.

In the beginning we focused a lot on the infrastructure which could provide such services.

Focusing only on infrastructure blurs the actual picture, draws your attention to implementation details, and you risk to end up with a maze of interconnections where it is difficult to see who does what, why, and when -> lesson learned!

Two themes are of utmost importance:

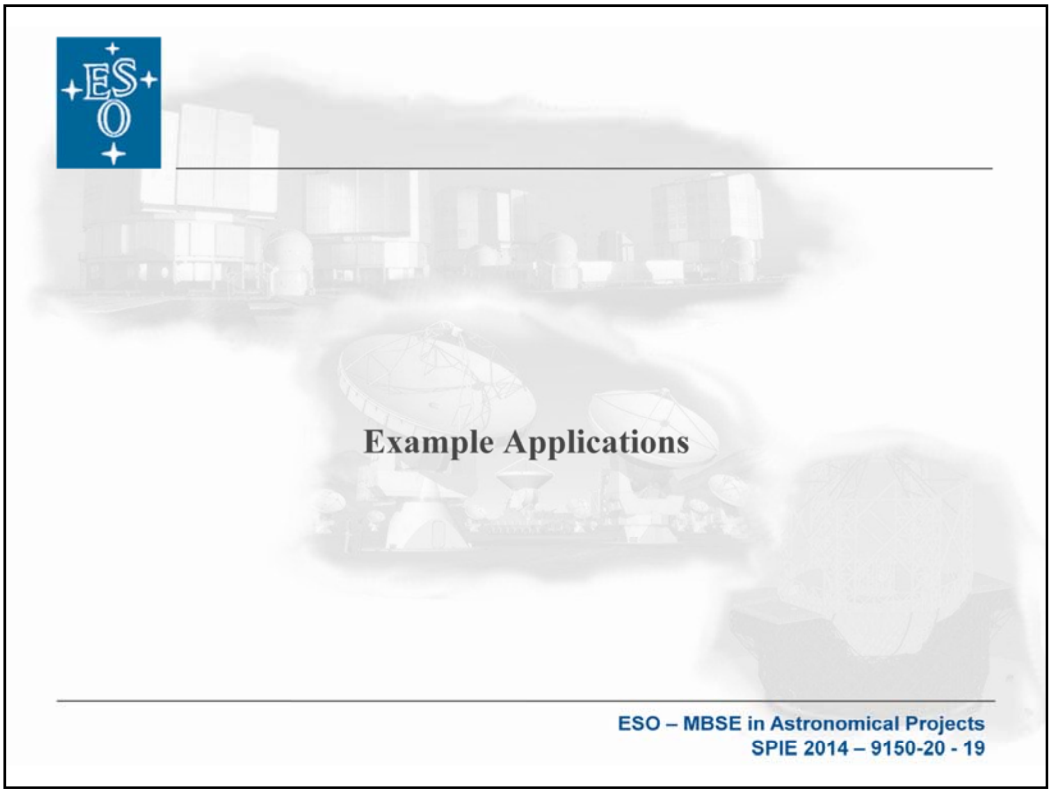
• Give structure to the maze by enforcing a well defined **architecture** with its principles and rules

• Explicitly define who does what, why, and when by defining the relevant **system states** and how to control them. Software states play a minor role in the end.

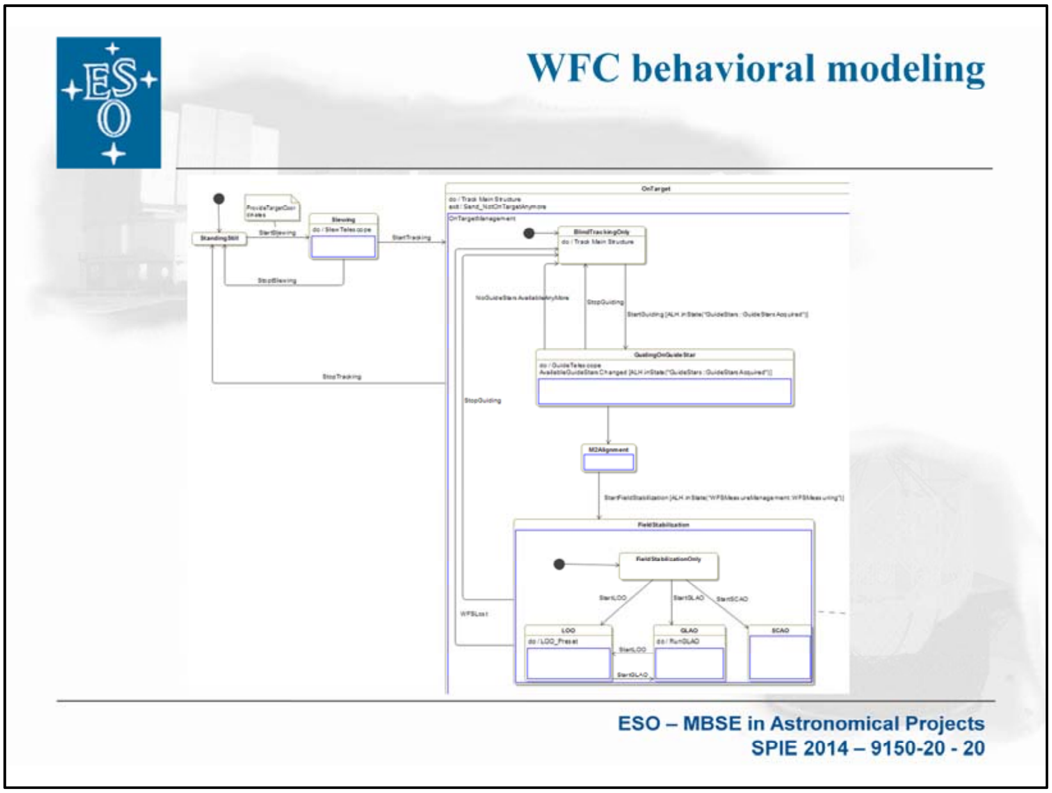
This will give us the means to provide a control system which is actually able to implement effectively the wave front control strategies devised by our system engineers.

The underlying infrastructure has to be selected in accordance with performance requirements and providing enough flexibility to implement

such an architecture.



I will show now a few examples extracted from real projects at ESO, where MBSE methodologies and tools have been or are being used and have brought tangible advantages.



For the purpose analyzing an ELT wavefront control (WFC) scenario.

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, the figure 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.

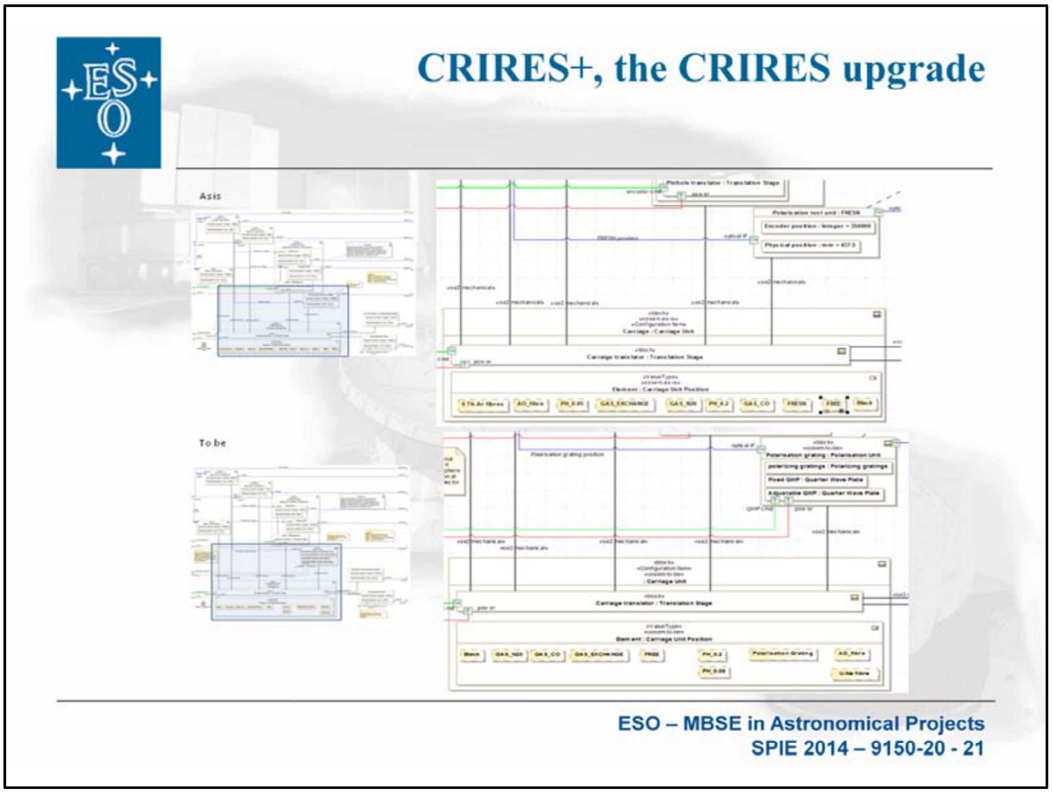
Simulation capabilities have been very useful:

- Discuss what happens when

- Validate concurrency and identify deadlocks

- Calculate time budgets

- ……



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.

An upgrade project is in progress to improve performance in the wavelength coverage, add a new polarimetric capability and modify a number of other subsystem.

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).

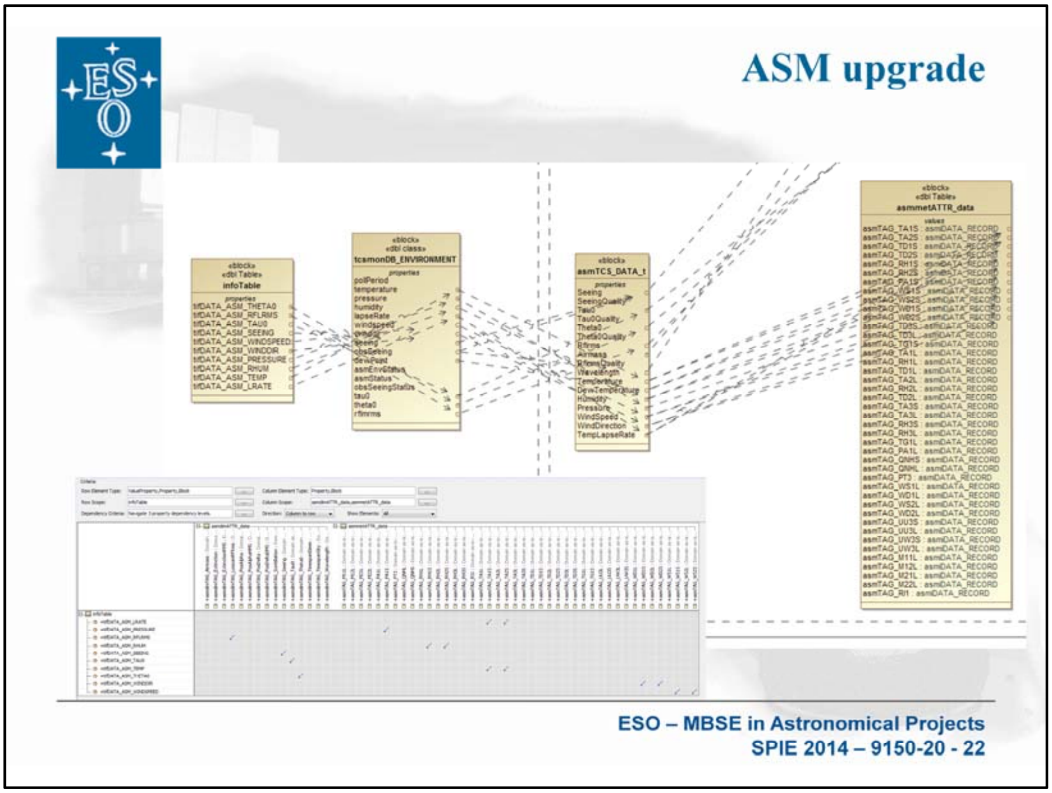
In the interest of efficiency, we limited this model to describing in detail only those parts of the system that shall be impacted by the upgrade.

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 led 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.

The figure gives an example of *as-is* and *to-be* models, in this case for the calibration selector subsystem.



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.

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.

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.

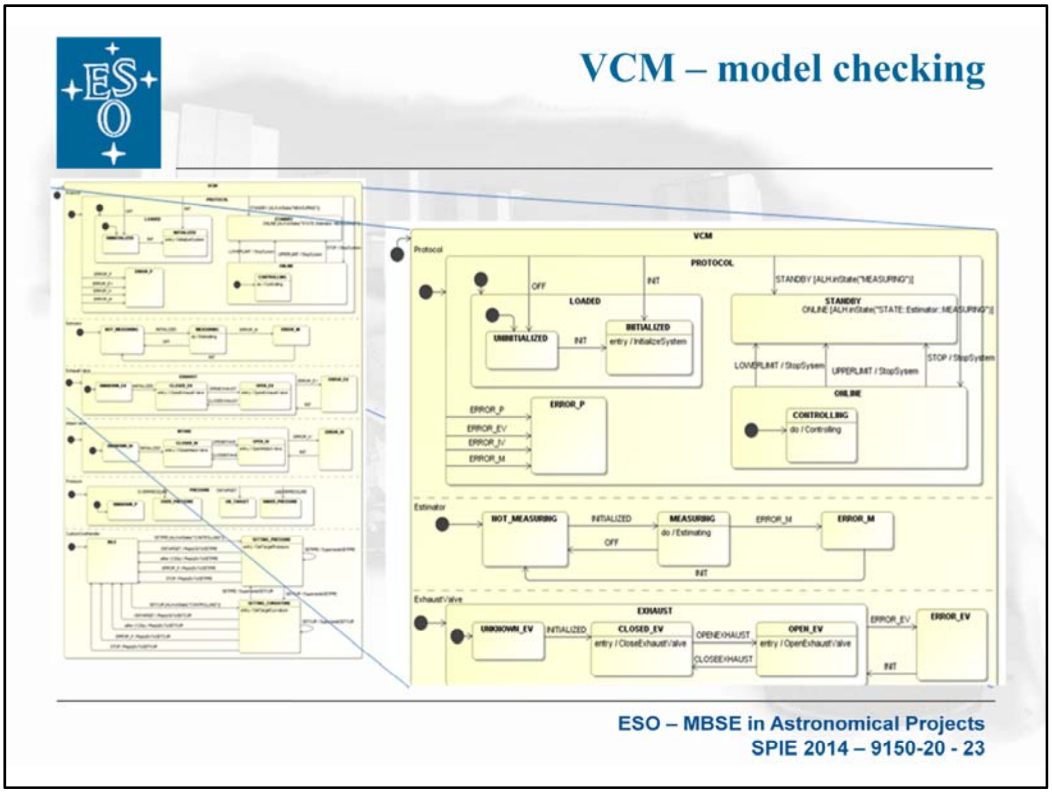
Since the information produced by the ASM is going through several layers before reaching the astronomical instruments (like data collection and processing and the telescope control interfaces), it is not trivial to identify what is really used in the instruments and what could be the impact of changes.

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 the figure 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 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.



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 got damaged 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.

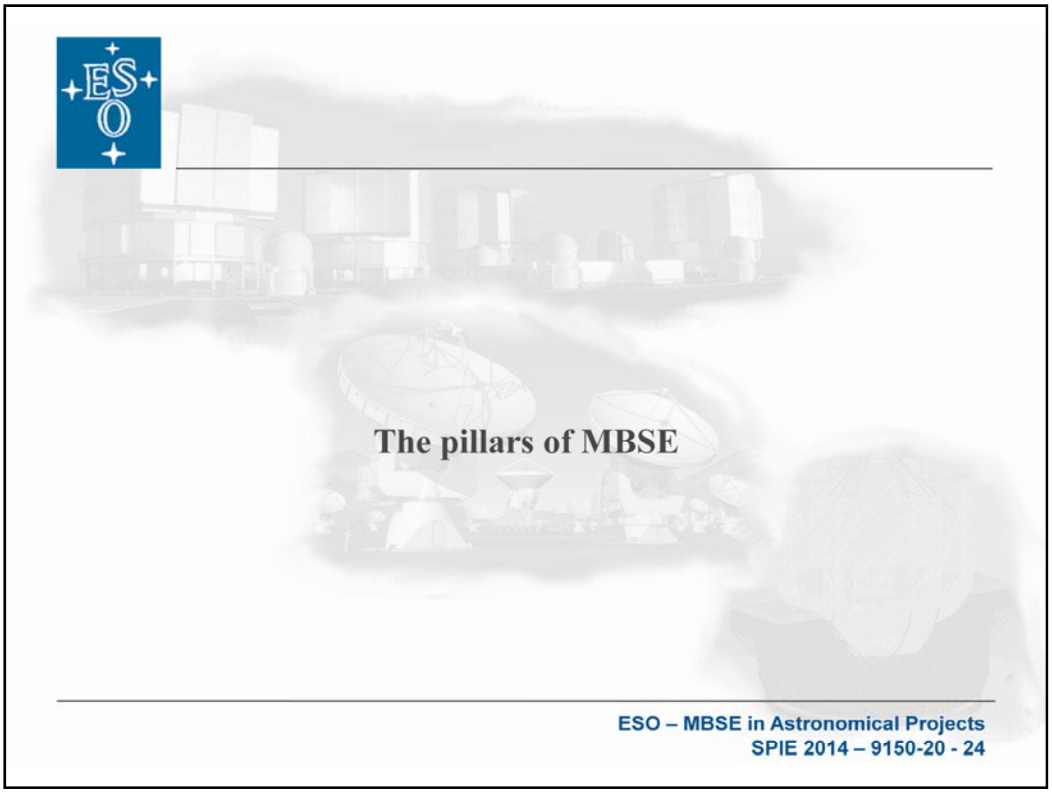
A Statecharts model of the new VCM control software for the Star Separator was created and validated using a Java Pathfinder model checker. The transformation from the Statecharts model to the Java code compliant with Java Pathfinder has been performed automatically using our COMODO tool.

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).

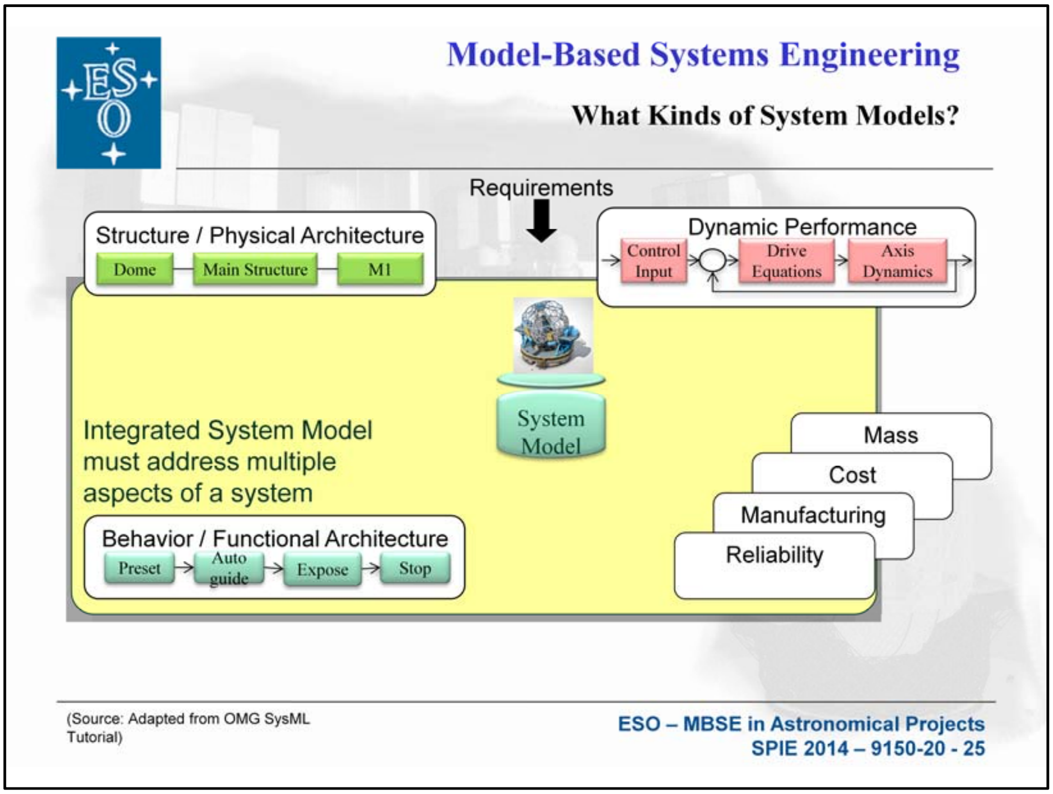
The model checker will try out all possible sequences of Statecharts events and stop in the case the CONTROLLING substate becomes active while MEASURING is not.

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.



- Modelling Languages
- Modelling Methods
- Modelling Tools



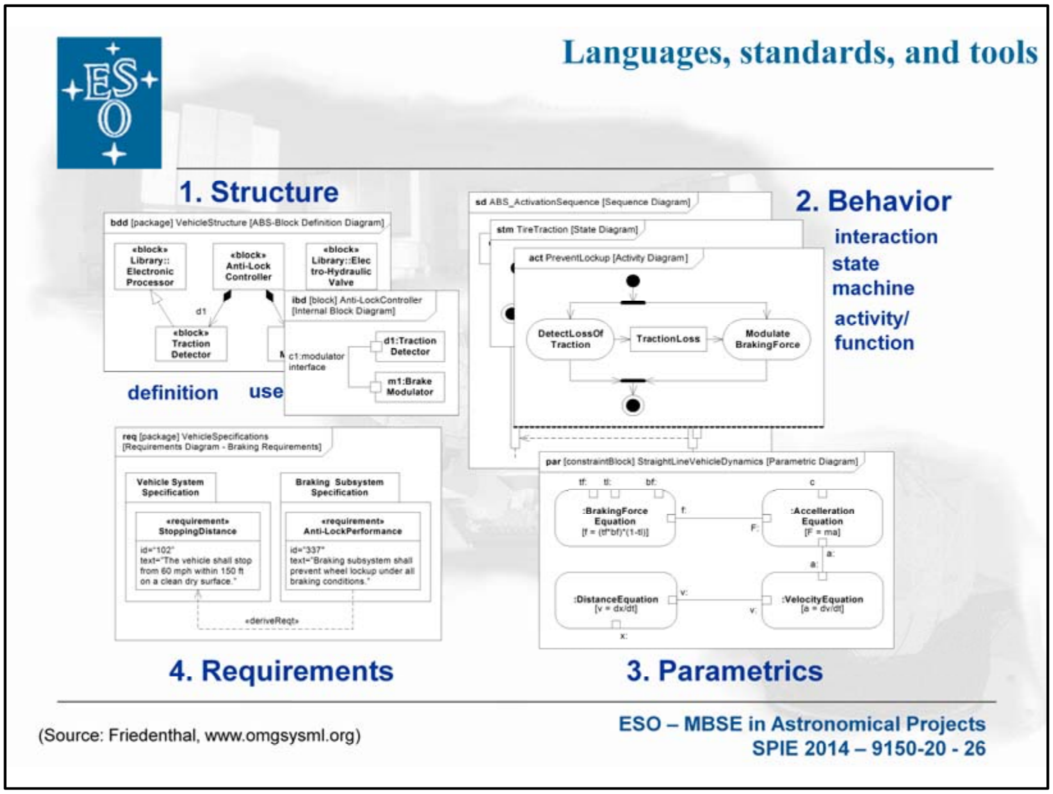
The integrated system model must be able to address multiple aspects of the system, allowing to build relations among them and navigate along the relations, starting from the requirements

... Physical, behaviour, dynamic and many other....

Different stakeholders are comfortable with different languages and different tools.

The challenge is to make everybody comfortable when using the model and avoid the feeling of being lost in a maze of unfamiliar things.

It is necessary to be able to recognize elements and navigate/modify the model intuitively.



The first thing we need is a common modeling language that can be used as a glue, also allowing the linking/referencing/analyzing of information coming for the many different models already used in the specific engineering disciplines .

SysML, the System Modeling Language, has been designed with this purpose, providing the basic semantics to model interdisciplinary systems.

A **system architecture** or **system*s* architecture** is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures of the system.

The restriction to and the highlight of the areas of interest of the different stakeholders is supported by the concept of Viewpoint.

Viewpoint is a systems engineering concept that describes a partitioning of concerns in system restricted to a particular set of concerns. Adoption of a viewpoint is usable so that issues in those aspects can be addressed separately.

A good selection of viewpoints also partitions the design of the system into specific areas of expertise.

Viewpoints provide the conventions, rules, and languages for constructing, presenting and analysing views. In ISO/IEC 42010:2007 (IEEE-Std-1471-2000) a viewpoint is a specification for an individual view. A view is a representation of a whole system from the perspective of a viewpoint.

A view may consist of one or more architectural models. Each such architectural model is developed using the methods established by its associated architectural system, as well as for the system as a whole.

The modeling language has to be associated with a definition of standards defining precisely the meaning of the various modeling languages.

Last but not least, the success of the adoption of any modeling language is strictly linked to the support provided by the tools to be adopted:

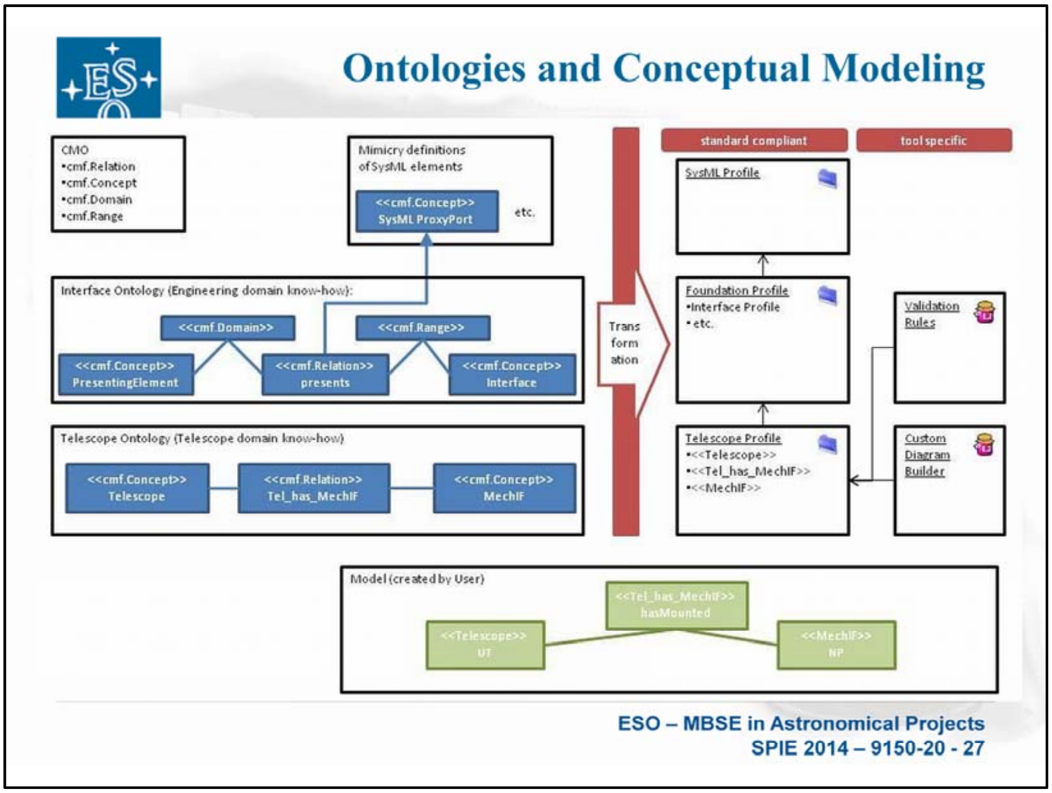
- Only a tool can enforce the adherence to standards

- Both newcomers and experienced users must feel comfortable in the usage of the tools, that must be intuitive and convenient. Fighting with the tools while also learning the language and the standards is a daunting experience.

SysML (and the underlying UML) are complex and too many choices are offered to modelers.

More over modelers are coming from different engineering disciplines and might not immediately understand the language of their colleagues.

Slide borrowed from Sandy Friedenthal from Lockheed Martin



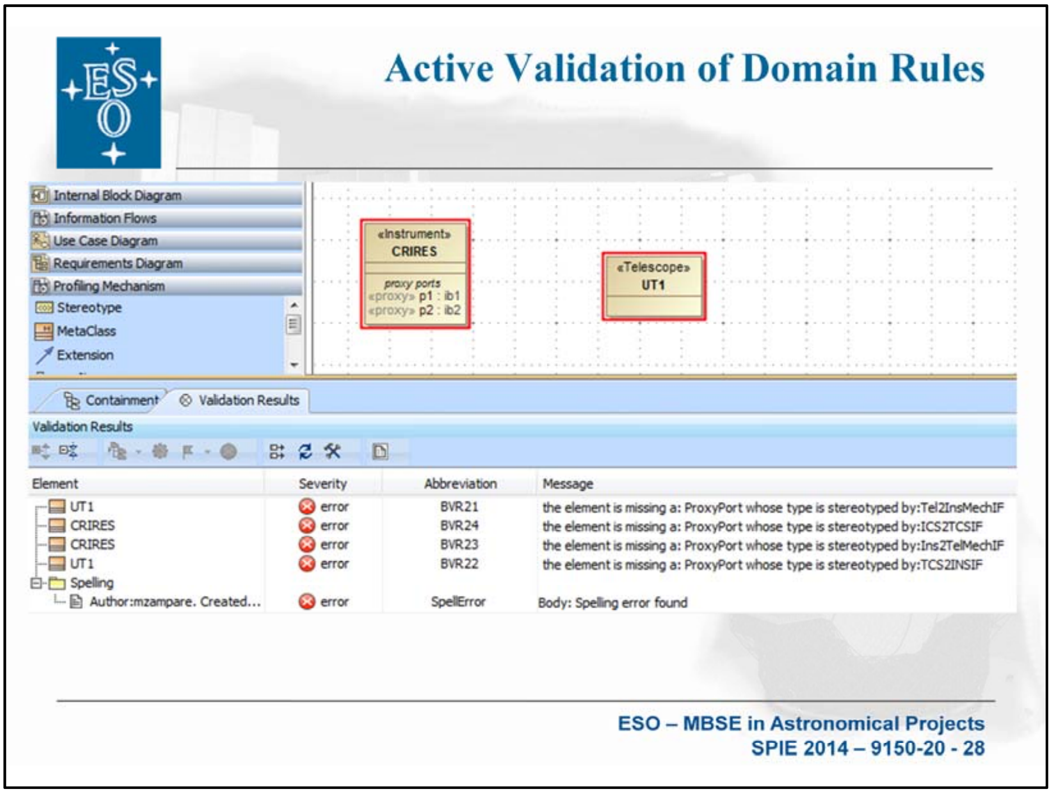
A good solution to this problem is to capture the specific languages and business rules in a way that allows the different stakeholders to express they concerns in a domain specific language, exposing at the same time a specialization and a restriction of SysML tailored to the particular use case.

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*.



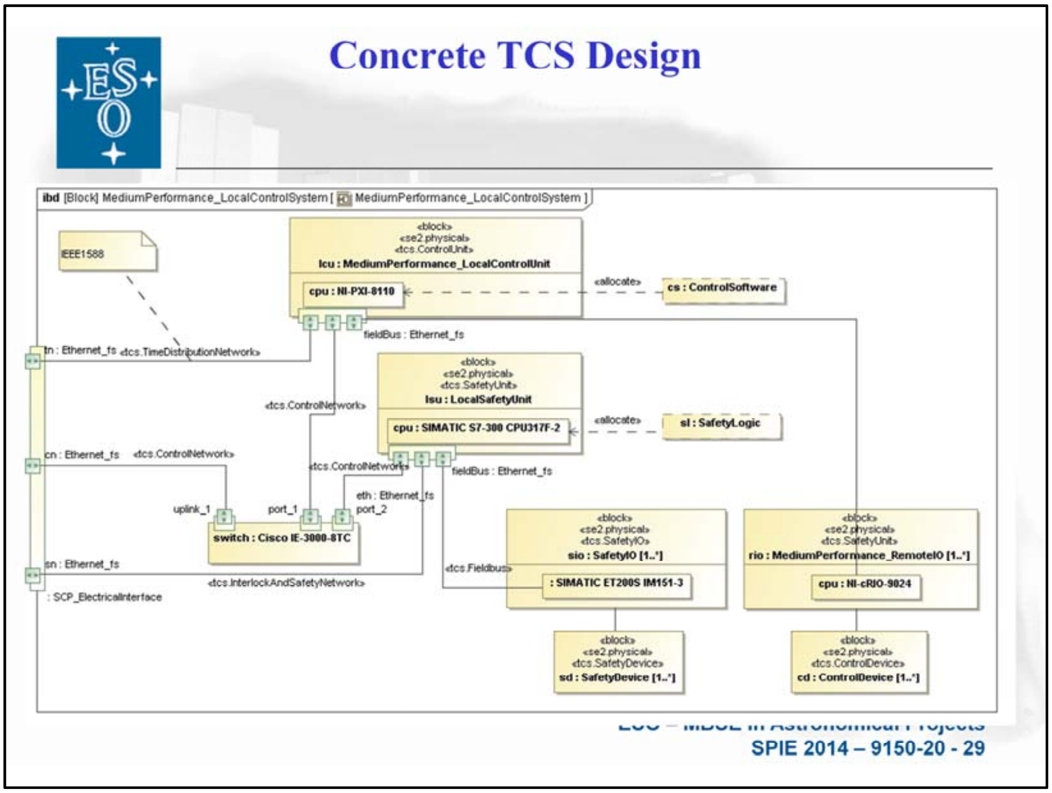
Some modeling tools (MagicDraw among them) offer a specific support for this methodology.

They allow, on top of what SysML offers to specialize the language, the possibility of creating plug-ins in order to:

- Specialize menus and user interface
- Create specific diagram types, restricting the usage of features
- Define modeling rules and provide immediate visual feedback to the modeler when modeling rules are being violated. The customized validation rules are directly pointing the modeler to a problem while creating the model.

In this example, we have just started modeling an instrument to be mounted on a VLT Unit Telescope and the modeling tool tells us that, according to our ontology, we need to have a specific mechanical interface between the two.

This allows to make sure that requirements are satisfied and helps the engineer in his work.



This diagram shows the model for the physical structure of a medium performance local control system for the E-ELT.

You can see the physical components of the system and network connections.

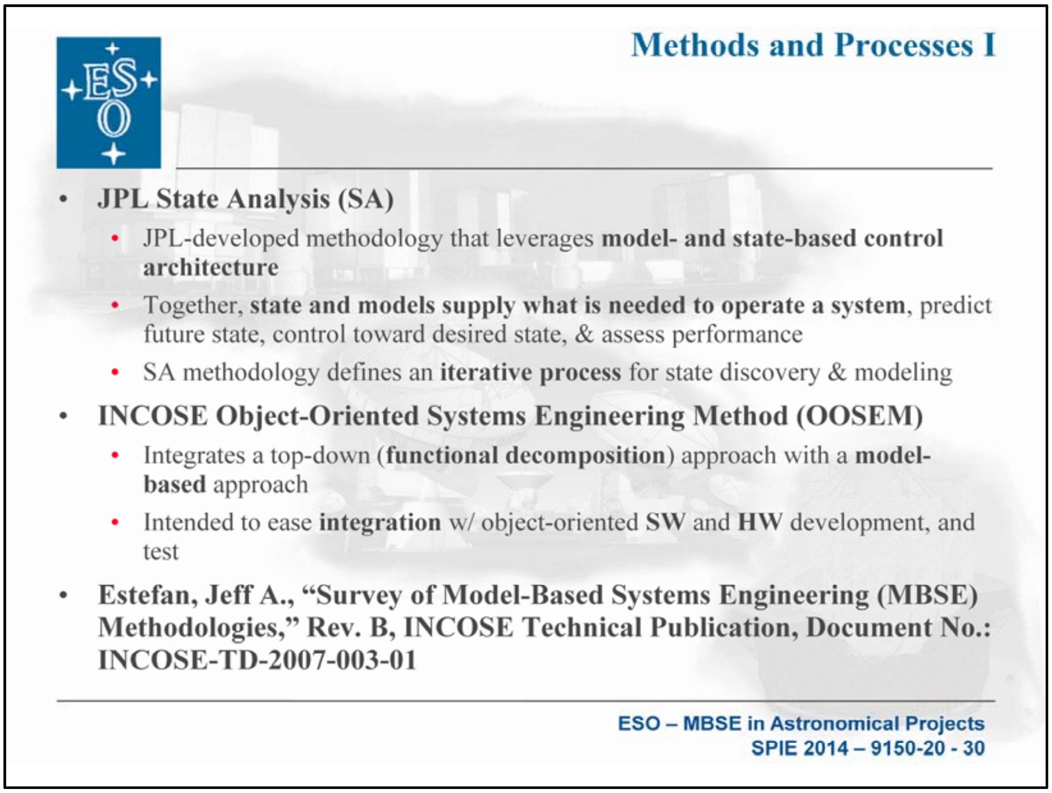
For example, you have a number of field devices connected to a remote IO module (specifically an NI Compact RIO).

This is connected to a processing unit on which the control software logic is allocated.

In parallel you see the independent interlock and safety logic system (based on a Siemens PLC).

The Cisco router allows to protect the safety network from the normal network traffic: it is possible to trace safety requirements and verify that they are satisfied by this architecture following the allocation chain.

The physical components are taken from a catalogue, where you can have information about price, power consumption, heat dissipation or any other parameter of interest. The model can then be used very easily to calculate the total cost of the system or the total power consumption. These can be compared with requirements to verify if we are within the budget envelope.

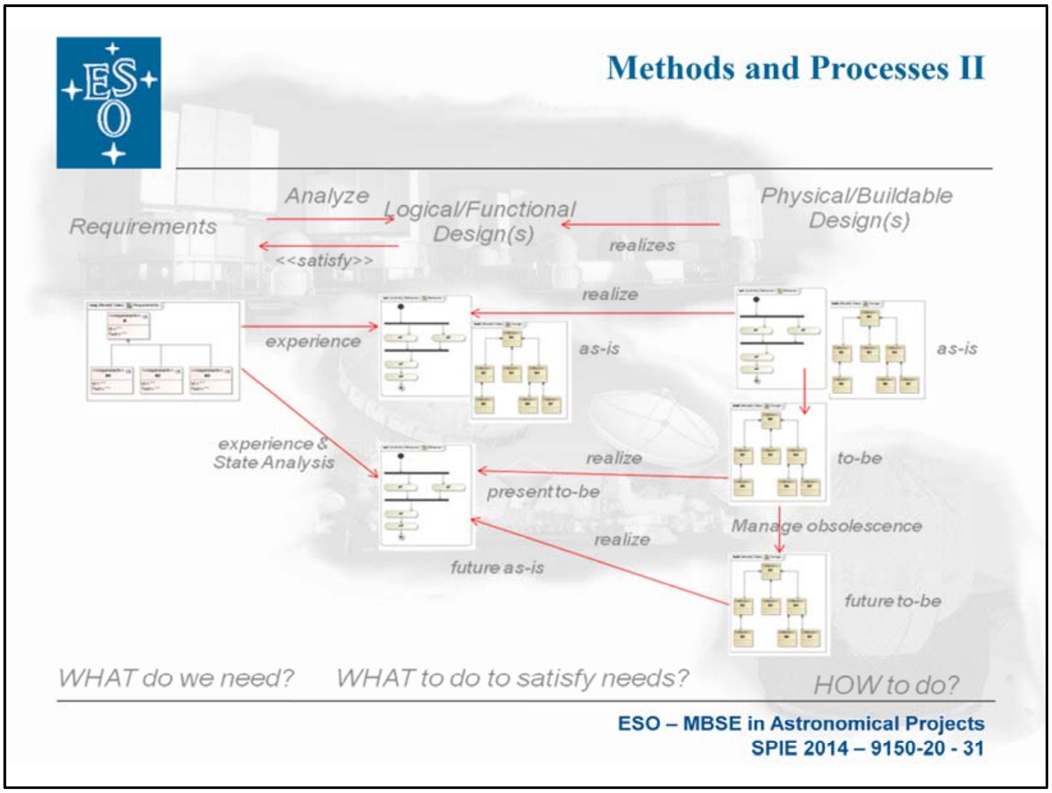


The other fundamental pillar of MBSE consists of the methods and processes that guide through the lifecycle of the system in the analysis and in the definition of the architecture.

At ESO we rely now on two main stones:

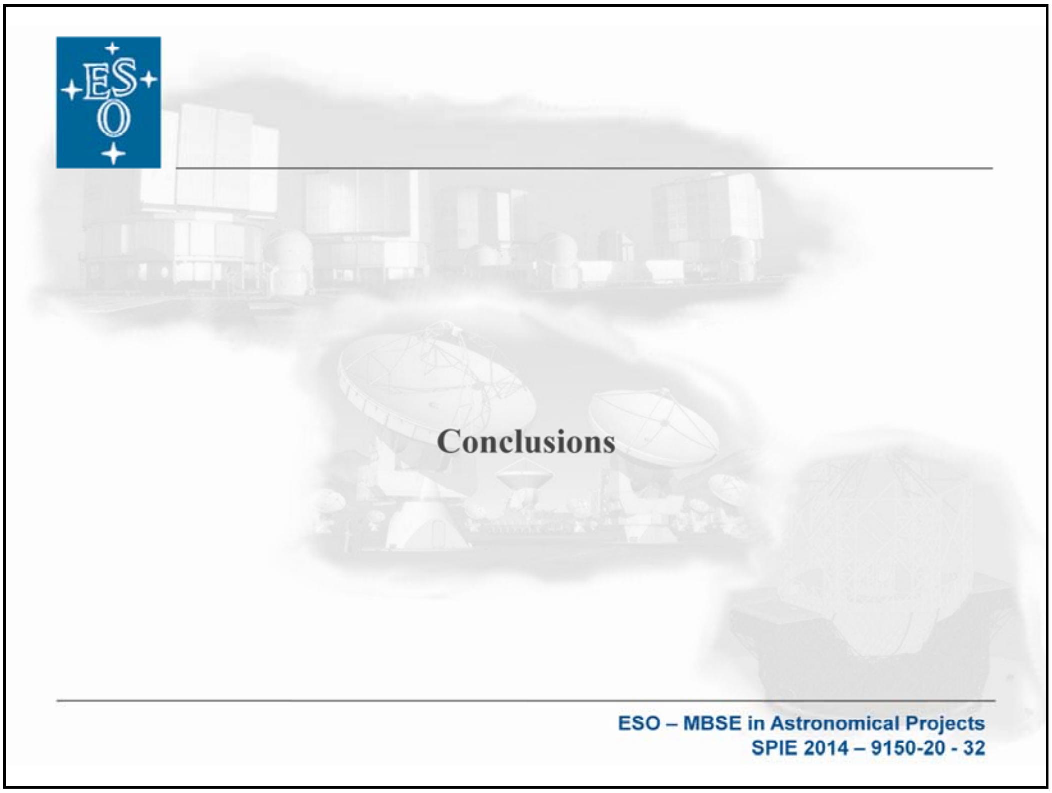
The **State Analysis** (SA) methodology is targeted to the control related domain, and focuses on behavior. It has been developed by NASA's Jet Propulsion Laboratory (JPL) since 1990. The methodology, which is founded on a state-based architecture and goal-based operation, defines a process for identifying 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 for SA in collaboration with the JPL.

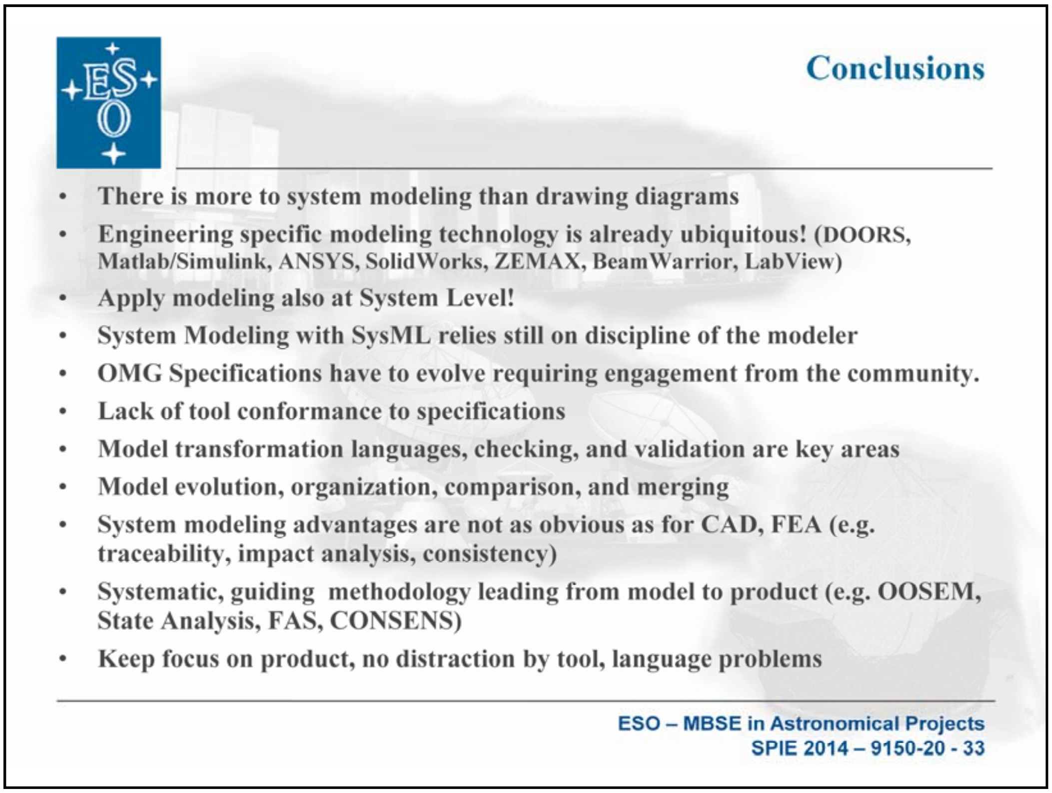
The **Object Oriented Systems Engineering Method** (OOSEM) integrates top-down functional decomposition with a model-based approach that uses SysML to support the specification, analysis, design, and verification of systems. OOSEM is intended to ease integration with object-oriented software development, 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.



OOSEM is very useful in analyzing a system to be implemented (to-de) taking into account what already existing (as-is) and also managing obsolescence (future-to-be).

This is very important in our domain, where we have to build new telescopes and instruments (to-be) based on our domain specific experience (as-is) coming from previous projects but also keeping into account the obsolescence problems emerging by long life expectancy of the systems (future-to be in 15 years from now).


Conclusions
ESO – MBSE in Astronomical Projects
SPIE 2014 – 9150-20 - 32



Summary: We are close an yet so far!

Once we have realized that there is much more to system modeling than drawing diagrams, we have started to grasp how much MBSE can in principle help us in getting faster and cheaper to a good quality product.

Engineering specific modeling is already used in several areas to help engineer producing specific deliverables, not just documentation.

Making nice diagrams that can be understood by several stakeholders helps in communication and in understanding the system, but this is just a first step and might not pay off the cost of imposing a methodology and tools to apply modeling at system level.

Additional benefits have to be demonstrated

- Simulation, validation, model transformation
- Reuse of design elements
- Generate multiple artifacts (e.g. documentation, code)
- Model transformation allows using capabilities of different tools
- Gain in correctness, consistency, time, money

Forcing usage is not worthwhile without obvious benefit.

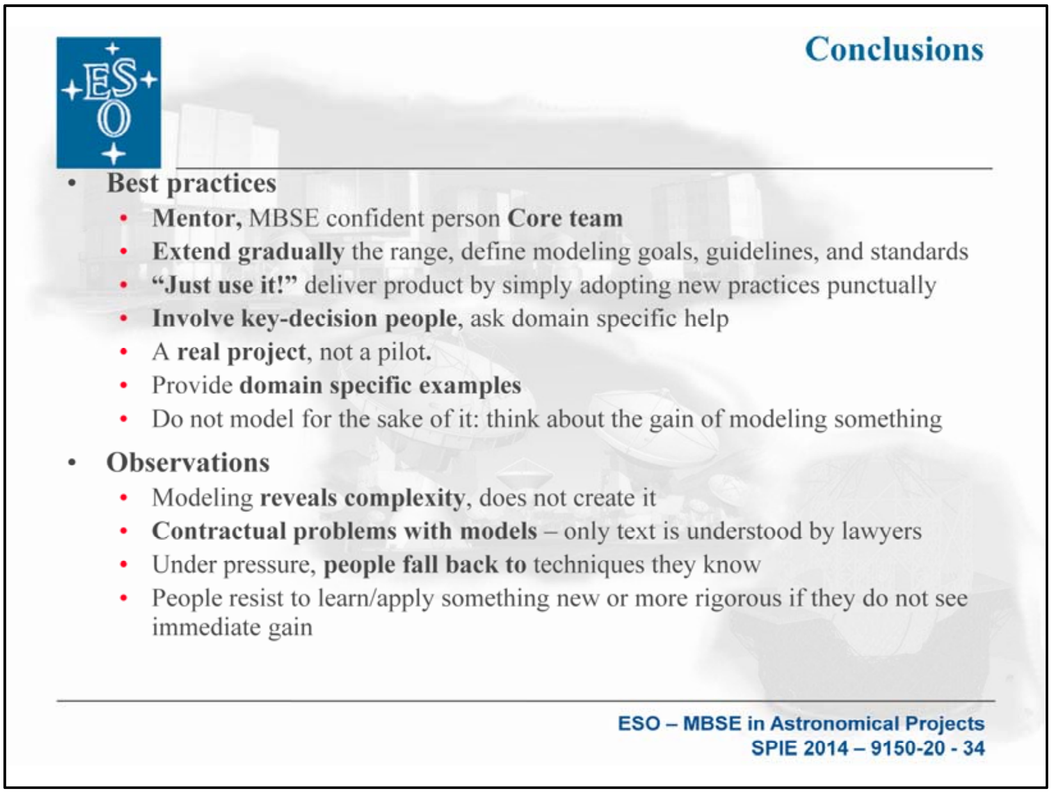
Specific modeling technology went down the same path. Think of CAD tools: from drawing to automatic production of the designed parts.

Vendor specific tools and languages are ahead the standards. They have full control over semantics but we would be locked in.

What does it mean if you draw two boxes and a line between them?

Example: nested ports in SysML. Ports have no concept of usage.

So, what does it mean if you conjugate a nested port in a diagram?



We can conclude with some best practices, that can be summarized by saying that it is not sufficient to say that we adopt the technology, provide a couple of examples, may be some formal training and then leave the people alone.

A core team shall use modeling in a project and then be available to spread the knowledge by helping the others, being aware of problems and advantages.

"True believers" are necessary to try to push the system modeling limits and find all sorts of problems in tools, spec, documentation possibly before the users find them.

Such exploration is necessary to provide guidance to the users, so they don't end up in a dead end, get depressed, and abandon.

What we shall do is…..

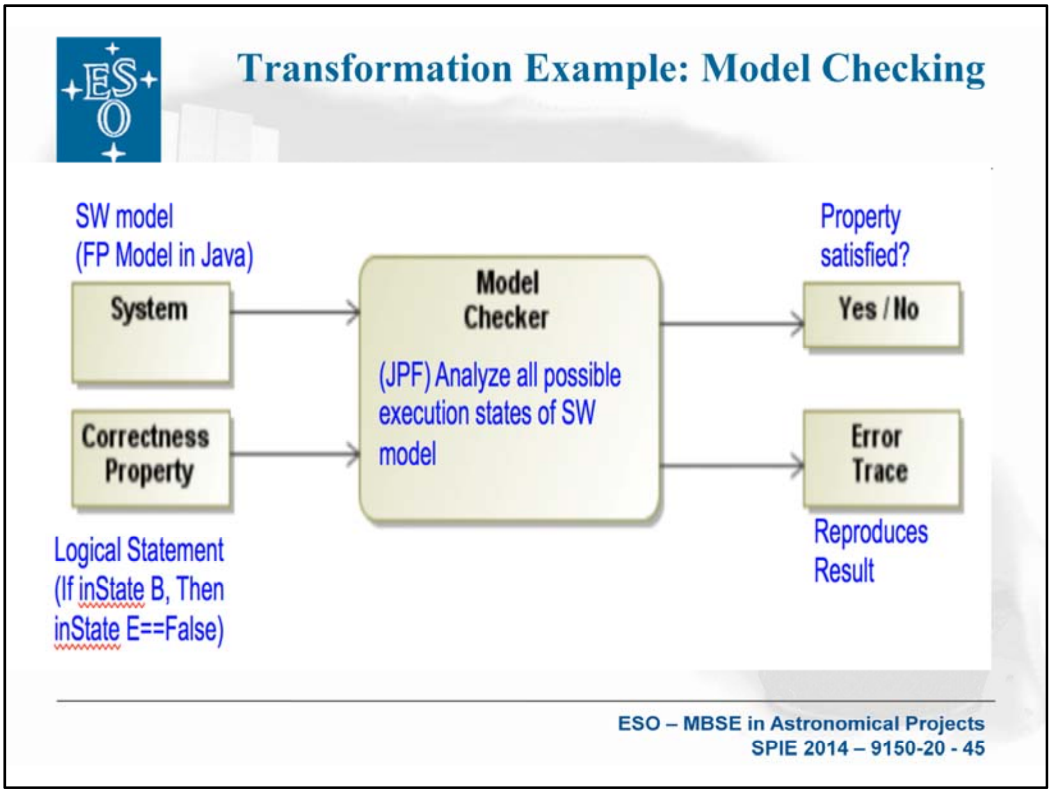
… Best practices …

Do not model for the sake of modeling, but model what is useful because several people need to understand unambiguously, or because it has consequences that need to be traced, or because there are several artifacts that can be produced from a single source and have to be consistent.

It is also important to keep in mind that….

… Observations …

Questions (& Answers)


Transformation Example: Model Checking
SW model
(FP Model in Java)
System
Correctness
Property
Logical Statement
(If inState B, Then
inState E==False)
Model
Checker
(JPF) Analyze all possible
execution states of SW
model
Property
satisfied?
Yes / No
Error
Trace
Reproduces
Result
ESO – MBSE in Astronomical Projects
SPIE 2014 – 9150-20 - 45