

Extending the System Model

**Mauro Pasquinelli, Luis Molina-Tanco, Arcadio Reyes-Lecuona
and Michele Cencetti**

Abstract This chapter briefly reviews the state of the art in existing system modelling practice in support of project activities that span a product’s lifecycle in different industry types (e.g., large series, small series, one-of-a-kind). Issues of collaboration, gaps in supporting the entire lifecycle and the advantages of defining (and sharing) semantics are discussed. The benefits achieved through the use of models to maintain control of system consistency are described, along with examples of the requirements for using this approach in practice and the potential impacts on company workflow. The maturity and expected advantages of known solutions and proposed extensions to current practices are also described.

Keywords Model-based systems engineering · Product life-cycle · Realizations modelling · Model-driven engineering modelling · Analysis and simulation

1 Introduction

Modelling can be defined as the definition of systems, processes and/or associated methods. Modelling requires dedicated processes, controls and resources. The modelling approach is not necessarily efficient; the associated effort may be lower

M. Pasquinelli (✉)
Domain Exploration and Science Italy, Engineering,
Thales Alenia Space, Turin, Italy
e-mail: mauro.pasquinelli@thalesaleniaspace.com

L. Molina-Tanco · A. Reyes-Lecuona
DIANA Research Group, Dpt. Tecnología Electrónica,
ETSI Telecomunicación, University of Málaga, Málaga, Spain
e-mail: lmtanco@uma.es

A. Reyes-Lecuona
e-mail: areyes@uma.es

M. Cencetti
Mission Operations and Training, ALTEC, Turin, Italy
e-mail: michele.cencetti@altec.space.it

or higher than the related savings or earnings due to improvement in quality, reduction of data exchange effort, reduction of programmatic and technical risks, prototype cost savings, easier feedback from stakeholders and provision of additional services regarding the physical or digital good produced.

The main objective of Use-it-Wisely (UIW) is to enable innovative continuous upgrades of high-investment product-services (see Granholm and Groesser in Chapter “[The Use-It-Wisely \(UIW\) Approach](#)” of this book), which requires:

- (1) Customer involvement, including providing the required information, receiving and capturing their feedback and anticipating their needs (one of the seven challenges identified in Chapter “[The Challenge](#)”).
- (2) Understanding the customers’ needs and transforming them into valuable innovative solutions through an adequate ideation and creativity process (see Chapter “[Complexity Management and System Dynamics Thinking](#)” for details).
- (3) An industrial strategic approach to analyse, plan, simulate and anticipate the impacts of the upgrade at the company, market and environmental levels (see Chapters “[Complexity Management and System Dynamics Thinking](#)”, “[Managing the Life Cycle to Reduce Environmental Impacts](#)” and “[Collaborative Management of Inspection Results in Power Plant Turbines](#)” for details).
- (4) Efficient and effective improvement of technical work to rapidly analyse updates and product innovation from as-required status to realized status through design, verification and post-delivery activities, to provide adequate engineering services for the customer and enter the design, verification or operations loop (the main purpose of this Chapter is to provide the means to respond to this need through a system-level neutral layer for all stakeholders).
- (5) Collaborative work between the project teams, customers and project stakeholders, supported by adequate approaches (see Chapter “[Virtual Reality and 3D Imaging to Support Collaborative Decision Making for Adaptation of Long-Life Assets](#)” for a potential solution for this need).

The complexity of these elements can be managed through modelling. This Chapter analyses modelling methodologies in the context of innovative upgrading of complex technical systems (e.g., space, airborne, heavy machinery, naval or energy systems).

How can issues related to the technical management of complex systems be handled efficiently? This question has already been answered: Through systems engineering and model-based approaches. Systems engineering is defined by the International Council on Systems Engineering (INCOSE) (Wiley and others 2015) as “an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem [...]. Systems Engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems Engineering considers both the business and the

technical needs of all customers with the goal of providing a quality product that meets the user needs.”

Systems engineering is not a self-standing activity; it is based on continuous consultation and collaboration with the other technical disciplines and with the program/project-level management. In parallel with this, system engineer activities rely on support from other SE activities for projects in different lifecycle phases.

Hence, an effective system approach relies primarily on the knowledge of those working on the team, a well-defined process, the collaboration, and the availability of required information.

Application of the systems engineering approach has led to the definition of various standards and methods to support all the perspectives that characterize a project. Different standards have matured to support systems engineering activities and reduce errors related to information exchange between different environments.

The model-based approach is a consolidated method to involve all the technical disciplines and relies on modelling to manage complexity and improve the effectiveness of the conception, definition, verification or operational activities using appropriate tools to improve efficiency.

Model-based approaches at the system level that replace or sustain the traditional document-based approach are applied or planned in many fields, such as military, space, transport, healthcare, robotics and telecommunications. This is revealed by the wide variety of universities, agencies and companies that are interested in the field and participate in projects and conferences worldwide (e.g., IEEE Systems of Systems Engineering Conference, INCOSE International Workshop and IEEE Systems Conference).

There are benefits to application of a model-based approach despite the limits to its current scope. The development of a transversal application that profits from modelling as much as possible throughout the entire project or product lifecycle and that unifies different disciplines and those beyond the company boundaries is still an open field of innovation.

This Chapter analyses some of the fundamental aspects of reaching such a vision. Section 2 provides a brief overview of the main types of modelling. Section 3 provides an analysis of the practical implementation of some models, proposes extensions and changes to available models and describes visions for the future before summarizing the conclusions in Sect. 4.

2 State of the Art in System Modelling for Systems Engineering and Technical Simulation

Systems engineering is currently gaining an increasing role in the design process for complex products. System modelling is a multidisciplinary approach that addresses the development of balanced solutions for different stakeholder’s needs. This balance involves both management and technical processes, with the main aim of reducing the possible risks affecting the success of a project. Management activities

mainly address monitoring development costs, schedules and technical performance, ensuring that the project objectives are met. These processes are related to risk management and decision making activities. Some of the most important activities performed at different levels of system development are:

- Elicit and analyse stakeholder needs
- Specify the system
- Synthesize alternative system solutions
- Perform trade-off analysis
- Maintain traceability

Two of the most interesting and challenging phases are synthesizing alternative solutions and performing trade-off analyses. A clear understanding of stakeholders' needs is crucial because the decisions made during this early definition process can affect the effectiveness of the final product. It is extremely important to understand how the external systems, users and physical environments interface with the system to clearly define the boundary of the system and the associated interfaces. This process is often characterized by the definition of the functions and related non-functional requirements that must comply with the customer requirements (functional analysis), specifying their sequence and ordering. After the functional analysis is performed, development proceeds with the design and testing of components, providing feedback to the specification process. In this manner, the design evolves iteratively towards the definition of the final system solution.

During this process, it is important to clearly define the information flow from the stakeholder needs to the component requirements. The system representation often includes broad stakeholder perspectives and involves the participation of many engineering and non-engineering disciplines. A typical systems engineering team should include viewpoints from each of these perspectives. Teams from different domain-specific fields must work together in a complex environment in which all the disciplines are deeply integrated.

The complexity of the systems drives the definition of a system of systems (SoS) structure in which an individual element is part of another system with a higher level of definition. The appropriate management of system complexity has led to the definition of various systems engineering standards to support different perspectives on the same project. Reduction of as many data exchange errors as possible is one goal of the standards. An overview of some of the most relevant systems engineering standards is available from (Friedenthal et al. 2014).

System modelling aims to define the processes and components that characterize a product through the entire lifecycle and across different domains. The main objective of the modelling standards is the identification of a common language for describing physical system architecture, behavioural models and functional flow.

Model and data exchange are among the most challenging and critical activities during development, especially when different domain-specific tools must interact for data sharing. Different modelling approaches and protocols are currently available in the context of systems engineering. The XML Metadata Interchange

(XMI) specification is an example of such a standard for facilitating model data exchange. In the same manner, the model-driven architecture (MDA) paradigm addresses the definition of standards, ideally enabling the transformation between models and different modelling languages. These efforts address improvement of tool interoperability, modular modelling processes and re-use of system design products, reducing the time and costs of implementing defined components.

Interdisciplinary communication is essential in establishing stakeholder needs. The integration of system modelling environments and frameworks for technical simulation is often affected by the communication between domain-specific disciplines. Communication among those with different backgrounds is challenging but critical for the effectiveness of the developed system. The use of different tools, procedures and formats to model and analyse the same product must be properly coordinated. A common conceptual infrastructure can improve the effective exploitation of simulation environments to support system modelling, ensuring a seamless exchange of data across disciplines.

2.1 Model-Based Systems Engineering

The model-based systems engineering (MBSE) methodology is one of the most interesting approaches in the system modelling domain and shows promising capabilities for management of the phases that characterize a project. The application of the MBSE methodology to support the design of complex systems has been assessed through different research initiatives such as in Space Engineering, a domain characterized by a high level of complexity in which the number of products, people, disciplines and processes leads to an environment that is difficult to manage and control.

The increasing number of variables and stakeholders, often from different backgrounds, make the task of properly managing a complex product very difficult. MBSE provides the basis for a rational organization of work with respect to traditional approaches. MBSE has been defined as (Technical Operations International Council on Systems Engineering (INCOSE) 2007):

Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing through-out development and later life-cycle phases.

One of the key concepts in the MBSE approach is Architecting, which is strictly related to the process that drives the identification of design solutions starting from system objectives. This process is characterized by the analysis and technical simulation necessary to evaluate system performances. During this phase, systems engineering work is also affected by policies, principles, procedures, budgets, reviews and other activities. Under these conditions, the system design process can be potentially characterized by omissions, misinterpretations and inconsistencies

that can be the source of issues in subsequent development phases. The main aim of MBSE methodology is the reduction of such problems because they can affect the system performance or delay the expected time to market.

In industry, lifecycle management generally includes product lifecycle management (PLM) and the related concepts that must be considered. PLM can be defined as the process of managing the entire lifecycle of a product, from the initial idea to the subsequent phases of design, manufacturing, operative service and final disposal. PLM integrates people, data, processes and business infrastructures, building the product information backbone for industrial companies to fulfil their mission. The lifecycle management processes of different products are characterized by slightly different phases with different temporal extents and conventions though they are all conceived to organize the work from the preliminary steps to the more detailed ones. Figure 1 depicts the activities and related relationships that generally characterize the overall process, from customer needs to the final system solution.

In the last decade, large-scale system projects have been created using different lifecycle development models. There are no constraints on the development models used by organizations, academia or industry. They often use their own lifecycle patterns, but the most common lifecycle models are Royce's Waterfall model (Royce 1970), Boehm's spiral model (Boehm 1988), and Forsberg and Mooz's "Vee" model (Forsberg and Mooz 1992). Each defines the lifecycle differently, as shown in their conceptual representations in Figs. 2, 3 and 4. Such lifecycle representations are partially derived from patterns used to implement software products and the same approaches can be applied and extended to the development of one-of-a-kind complex systems involving hardware and software.

The definition of development processes through V-shaped diagrams allows for a graphical description of the overall process of system design and development. This representation can be used to model the same conceptual process at different levels of detail because the same structure can be adopted to define the whole system and single subsystems or components.

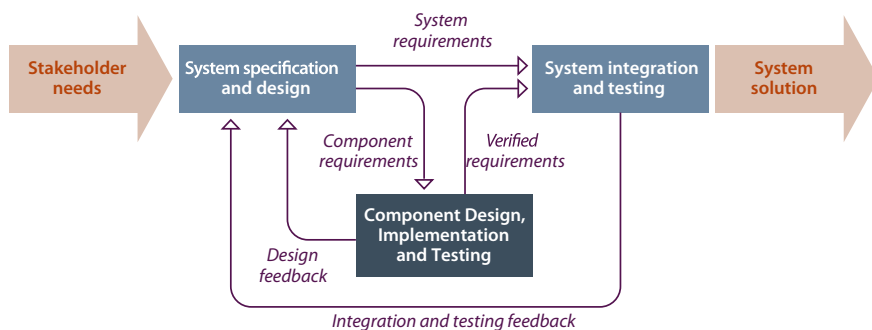


Fig. 1 Development process from customer needs to system solution

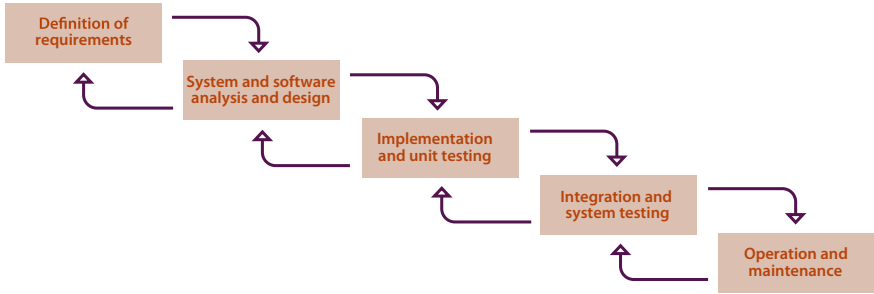


Fig. 2 Royce's Waterfall model (1970)

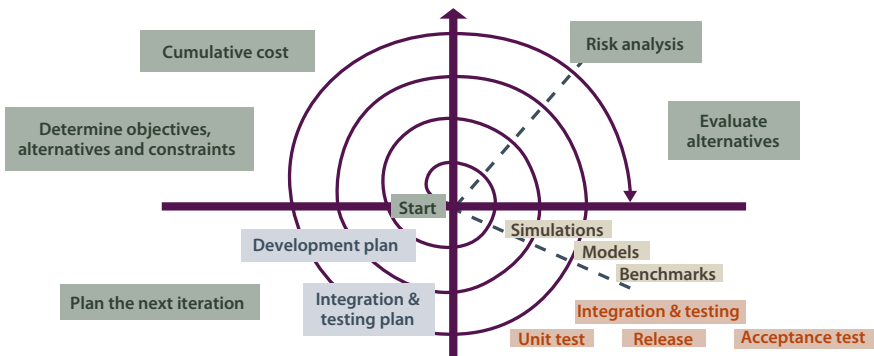


Fig. 3 Boehm's spiral model (1988)

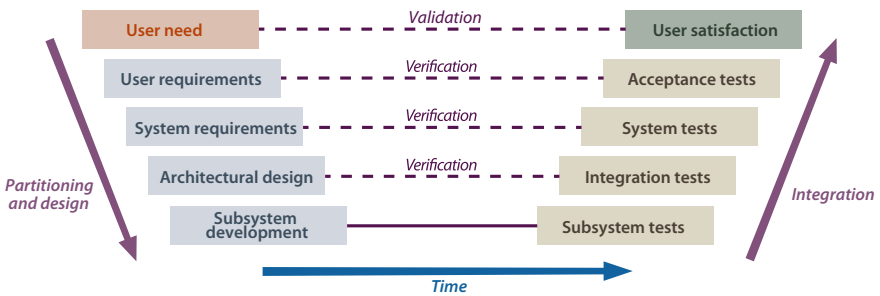


Fig. 4 Forsberg and Moog's "Vee" model (1992)

2.2 *Technical Analysis and Simulation: Languages, Methods and Tools*

As described in Pasquinelli et al. (2014), many types of models can be used in a model-based environment, and some are already widely used, especially in engineering disciplines. An initial classification limited to engineering activities may be organized into system-level models, engineering discipline models and collaboration models.

System-level models formally describe system-level views of system data (e.g., functional, architectural, behavioural, requirements). The Object Management Group standardized UML (Object Management Group 2015) (mainly for software) and SysML (Object Management Group 2013) (for systems). The xAF architectural framework (Rouhani et al. 2015) describes high-level (e.g., enterprise) architecture. Other examples include the ESA OCDT (de Koning et al. 2014) (for preliminary design) or VSEE (Rey 2013) (intended for entire lifecycle), the Thales ARCADIA approach (Roques 2016) and the TAS DEVICE model (Di Giorgio and Wiart 2012). CAD models define and maintain physical configurations, item arrangements, related interfaces and harness routing and are currently supported by many commercial tools.

Discipline-specific models are widely used in engineering for simulation and analysis. They represent a simplification of the real system from the perspective of a specific discipline. The geometry can be simplified for specific calculations and control. Continuous models can be discretized to solve problems using partial differential or differential algebraic equations. Software and logic models represent specific behaviours for implementation in system software or simulation of external operational entities.

Project collaboration models are also extensively used to manage workflow and change. Workflow models help define team tasks or work packages with associated input/output. Typically managed using PDM/PLM or corporate tools, they can sometimes be oriented to the formalization of contractual tasks rather than in support of daily work. Typically, such tools include authorization workflows and documentation management. For more technically oriented purposes, such models include the input/output definition from analysis and simulation and give control to a system architect/engineer for system analysis and simulation while gathering discipline-specific models. Change management models typically analyse relationships between existing models and can help provide an impact analysis in the case of a change.

Finally, optimization models typically connect different parametric models and enable finding the optimal solution according to objectives (Cencetti 2014).

As shown in Fig. 5, a centralized unique system model cannot exist because many models rely on its data and should be kept consistent. The system-level model could be a federation of models such as a SysML model or a Capella (Roques 2016) model for functional aspects (or an evolution based on semantics such as the

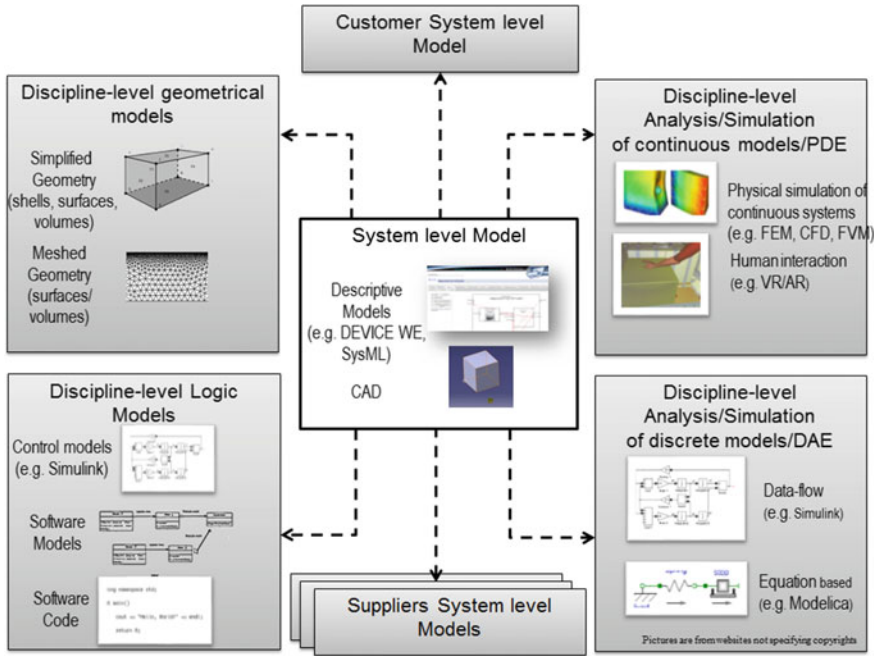


Fig. 5 Discipline-specific models rely on data and should be kept consistent

TAS DEVICE model) connected with the geometrical baseline (typical a CAD model).

The system-level information is mostly generated by discipline-level activities, providing analysis of requirements and evolution of the design in lower-level detail. Typically, discipline-level activities require a subset of the system model data and provide another subset of the system model data that is needed by other disciplines.

However, discipline-level models are not a subset of the system model. For instance, the geometry of a thermal model can be simplified with respect to a CAD model, excepting some items and including fictional items related to the simplification of the model for calculation purposes (especially in early phases and for some specific analyses). A mechanical FEM model includes more items than the CAD model meshes and has many properties that do not need to be shared. A software model could be mapped to system functions but it is not useful to share and maintain software-specific items (classes, protocols, etc.) in a central repository.

One typically overlooked aspect is the collaboration with different industrial tiers (i.e., customers and suppliers). In the case of model-based environments in industrial teams, a connection between models (with precise workflow and rule management) is essential for the highest profit from consistency control, clear flow of data between partners and control of impacts in the case of changes.

System/Architectural methods and initiatives are approached differently, from the SysML effort to provide a standardized language to other initiatives at company, project, or open communities levels (e.g., the DEVICE or Capella initiatives). The examples reported here were analysed during the UIW-project, leading to the study of custom solutions.

Systems Modelling Language (Object Management Group 2013) is a joint effort of the Object Management Group and INCOSE to standardize MBSE. SysML is a graphical modelling language with nine diagram types to model system requirements, functionality, behaviour and structure.

SysML (Fig. 6) has roots in Unified Model Language (UML), which is widely used in software and was designed to be exchanged with XML. SysML is a methodology-agnostic and tool-agnostic open standard, implemented by many commercial and free tools. The OMG SysML and tool vendors form a dynamic community and the standard and tools are frequently updated. This effort is an ongoing process that started in 2001; OMG SysML specification 1.0 was released in 2007, and the latest version 1.4 of the standard was released in September 2015.

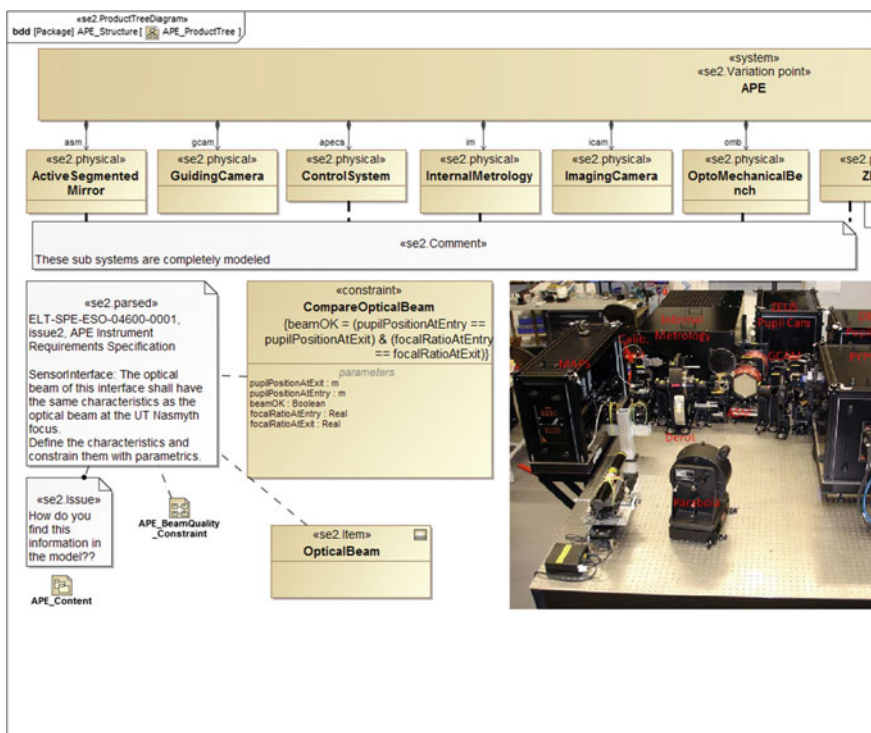


Fig. 6 A fragment of a structural SysML diagram (Karban et al. 2011)

Recent commercial tool vendor efforts have situated SysML at the centre of the MBSE approach (Fig. 7). Syndeia software for lifecycle management (Intercax 2016) claims to use the language to interconnect models between discipline-specific tools such as CAD, project management, requirement management, simulation, PLM and relational databases.

DEVICE (Distributed Environment for Virtual Integrated Collaborative Engineering) is a collector of Thales Alenia Space Italia internal research devoted to study, development, validation and proposal of new methodologies/tools to improve systems engineering and multidisciplinary collaboration since 2007 (Di Giorgio and Wiart 2012). This research was recently realigned with the Thales engineering environment deployed across all Thales business units and also partially deployed (for model-based system architecture tooling) since 2015 as the Capella open-source initiative under the Polarsys project (Blondelle et al. 2015).

The modelling portion of the DEVICE infrastructure is currently a customized conceptual meta-model that was conceived to be compatible with European standardization, e.g., ECSS-E-TM-10-23 and ECSS-E-TM-10-23 data models, which are not current standards but are meant to change significantly in the future. The end product is incrementally defined in terms of structure and behaviour with regards to

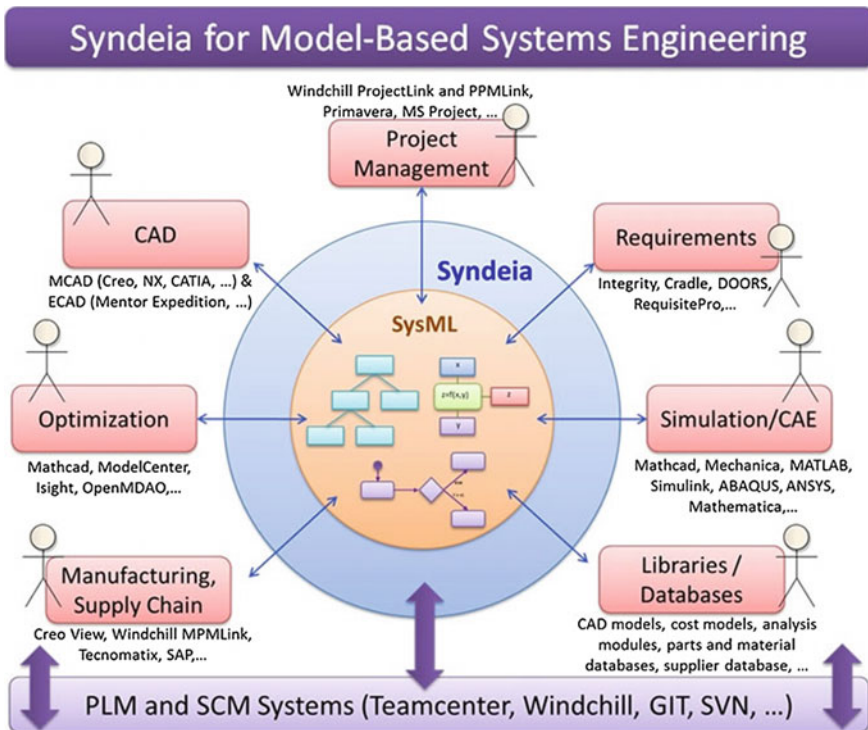


Fig. 7 SysML is in the centre of a tool-interconnection effort (Intercax 2016)

the lifecycle phase. Different design methods are allowed for logical and physical architectures, with relationships that allow precise semantics but do not create many different items. The defined semantics allow the product model to be used as a virtual model for simulation, allowing linkage with design tools, analysis models, test equipment and operational data.

The end product definition and verification are driven by requirements and reference scenarios, the requirements that are defined using models and are related to design items for automatic consistency checks. The scenarios consist of the definition of activities that will be performed in the in-flight utilization phase and during the production, integration, on-ground testing and logistics.

The UIW-project provided a cross-linked environment between the DEVICE research, typically validated in space activities and other domains and lifecycle activities that can enrich the existing models and approaches and inspire novel usage of standard tools such as SysML-based commercial tools or open source platforms such as Capella. The following section describes some of these results.

ARCADIA (Roques 2016) is a model-based engineering method for systems, hardware and software architectural design. It was developed by Thales between 2005 and 2010 through an iterative process involving operational architects from all Thales business domains. ARCADIA is the systems engineering methodology supported by the Capella tool (Roques 2016). This methodology was developed internally by the Thales Group and has been made open source. This methodology relies on several interconnected modelling levels:

- Needs understanding in operational analysis, i.e., an understanding of the operational environment that is independent from the existence of the system, and system analysis with objectives of defining the boundary of the system with respect to external actors and the system-level functions.
- Solution architectural design in terms of logical architecture, i.e., allocating the functions to logical components, physical architecture, i.e., defining how the system will be developed and built, allocating functions to hardware and software components, and detailing the interfaces, and end-product breakdown structure for managing industrial criteria and associating requirements and interfaces with configuration items

Figure 8 summarizes the ARCADIA Methodology. Other notable initiatives from the space field that have been used as references are the OCDT (<https://ocdt.esa.int>) and the VSEE (<https://vsd.esa.int>).

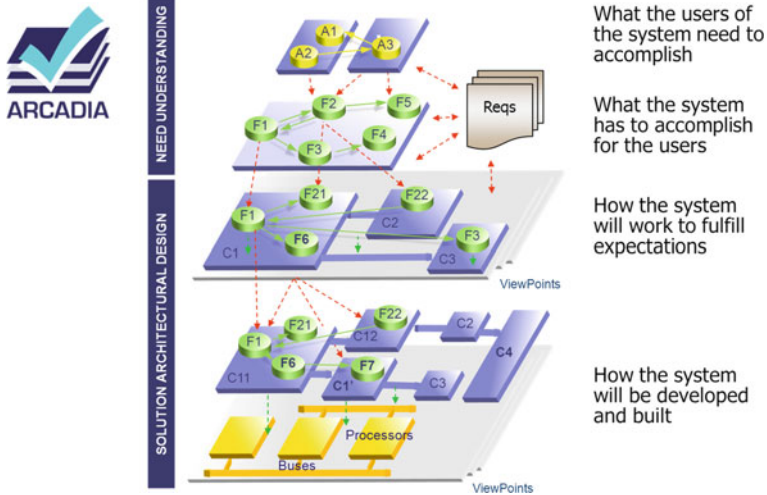


Fig. 8 The ARCADIA methodology (Roques 2016)

3 Extending the System Model to Cover the Entire Lifecycle

Three main gaps were identified in the ideal integrated methodology shown in Fig. 5: (1) integration of system modelling with simulation, (2) use of system modelling across the entire lifecycle and between different activities, and (3) tool limitations: security, data exchange, collaboration and user culture. A quick overview of such gaps is provided below.

Different issues can arise when system modelling methodologies are integrated in simulation environments. Development of the proper interfaces is strongly affected by the manner in which the integration is implemented within the overall design and analysis process. A clear understanding of the overall process and the related infrastructure can reduce issues that arise as development proceeds. A clear conceptual framework is fundamental to support modelling activities because it paves the way for effective exploitation of available resources.

Each of the existing methodologies is based on a specific data structure developed for specific applications, reducing the possibility of re-using the related environments within other contexts or domains.

The integration of simulation environments and system modelling frameworks can be approached in different manners depending on the final objectives and on the specific workflow that characterizes the company. Technical simulations can be used to investigate the product performance based on available data; the management of this information affects the integration architecture. The potential solutions can change based on the tools and required capabilities. Multidisciplinary analyses

can be supported using dedicated platforms to manage the results generated by simulation tools.

The exploitation of technical simulations is strictly related to the capabilities of external analysis platforms. System modelling environments can be used to store the representative project information whereas simulation platforms use such data to set up and execute analyses. The data exchange across different environments represents a challenging process because it is often difficult to integrate information from different sources in a straightforward manner. Currently, different tools support system modelling, but there is a gap between the available information and analysis capabilities (sensitivity analysis, optimization, uncertainty quantification, and parameter estimation). Object-oriented solutions can enhance the advantages of a unique environment for modelling and technical simulations and can reduce the efforts required for consistency verification when the data are exchanged directly with an external process manager. The gap between simulation tools and modelling environments can be mitigated if a common conceptual infrastructure is defined. Such integration can be realized only if the platforms and related methodologies are clear and well-posed.

The improvements related to a successful MBSE method on a system engineer/architect level can be jeopardized by an ineffective connection with discipline-level work, especially for data exchanged between different disciplines and managed for consistency at a system level.

SysML, ARCADIA, UML, xAF and other frameworks rely on a data model that was created for specific purposes and was not intended to serve for the entire lifecycle or for all engineering activities. Moreover, such methods (and related tools) were initially developed to support SE activities and not for an asynchronous collaboration between different team members using different tools. Recent advancements in such model-based methods and tools have attempted to overcome this limitation. However, the lack of semantics (for simulation use) is still an issue in many methods and it often necessitates ad hoc solutions, e.g., with a specific profile for the SysML. In such cases, the standardization is often replaced by vendor-level, company-level or project-level profiling.

Many different companies and institutions already rely on MBSE solutions, SysML-based (Spangelo et al. 2012) or custom (Di Giorgio and Wiart 2012), and almost all large enterprises rely on legacy systems that include relevant data for past and current projects.

Currently, the system modelling–simulation connection is an open area in which many improvements can be made, especially with the objective of an improved rapid response to the customer.

System modelling across the lifecycle

Systems engineering is often regarded as an approach to provide system solutions but after deployment other disciplines such as project management are more prevalent. MBSE has followed this view and focuses on the initial concept and

development stages, i.e., on the as-designed system rather than on the as-built system.

SysML, the main example of ongoing MBSE standardization, has a similar bias, lacking clear semantics to differentiate views of the system at different life-cycle stages. This limits the adoption of MBSE for the management of stages beyond concept and development. It also limits the mapping and interchange of information between tools across the life-cycle. One example is the configuration management of as-designed system components that can model design evolution versus configuration management of the as-built system components that can model system part maintenance and replacement. The link between the as-designed and the as-built statuses of a system is often at the frontier between tools and is often not explicitly managed in any of them.

Moreover, a typical Vee cycle should be supported by different types of tooling and methods for the relevant activities performed at each stage. In the space field, concept and feasibility studies conducted at the beginning of the typical life-cycle often rely on parametric models for early sizing and analysis to understand the system-level feasibility of the proposed solution.

In later phases of system and product definition (phases A/B in the European space standardization), a more complex industrial team is formed and more complex analysis is required. Moving towards the detailed definition and production phases, the overall consistency should be managed at different levels and the management of changes and requirements becomes more formal and controlled to assure the highest product quality and reduce risks. In serial production, the product and component variants and the ever-growing trend of customization increase the complexity. Modelling of product features, options and variants is not directly addressed in this chapter but their application is essential to any type of production, including one-of-a-kind, because they allow re-use of components that generates savings. The operational and disposal phases are typically supported by models to support the users, operators, maintenance and anomaly investigation teams (Fig. 9).

The analysis of the potential system- and product-level model-based activities is performed from a project lifecycle point of view and could be called “vertical”, viewed as a sequence diagram, with time progressing from the top to the bottom. There is also a horizontal perspective that should be considered as each activity runs in parallel to activities in other projects or even in other companies involved in the project. This issue has two main associated topics to consider:

1. Collaboration: in model-based systems engineering, the capability of IT tools to exchange models and synchronize them with limited effort by the user is essential for effectiveness and efficiency. Security issues should also be considered to avoid spreading sensitive company knowledge outside of authorized boundaries. Moreover, the use of common rules, object libraries and conventions is essential to assure an effective collaboration and reduce related risks.
2. Return of experience: experience gained in other projects or activities should be appropriately incorporated into the current activity. The MBSE approach can

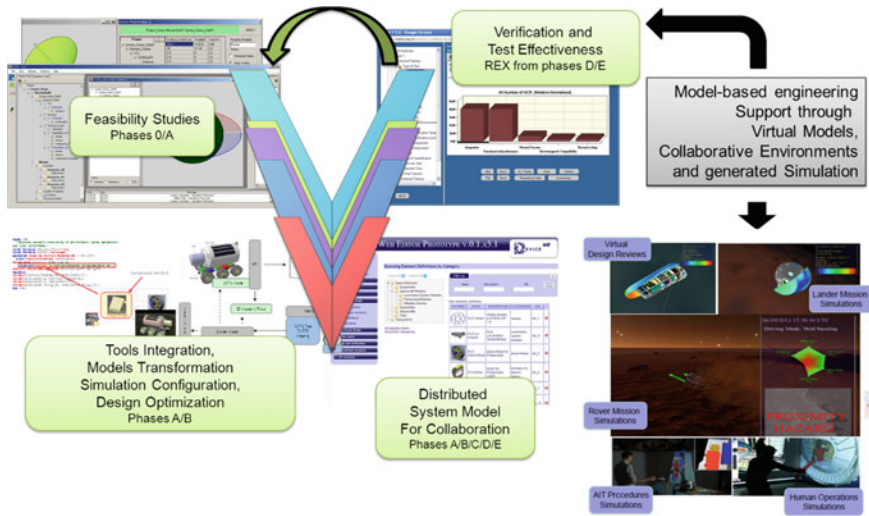


Fig. 9 Model-based usage across the lifecycle (Pasquinelli et al. 2014)

generate a large amount of data that, in contrast to the classic document-based approach, can be processed in a more effective manner.

Tool limitations: security, data exchange, collaboration and user culture

In many cases, the limitations are not in the methodology or languages but in the tools, toolchains and complexity of the information system networking in a multifaceted industrial scenario. There are three main factors that limit the seamless adoption of a tool by an end-user:

- **Overwhelming complexity:** if managing the data in the tool or understanding the user interface requires more time and mental effort than usual, the end-user may be reluctant to adopt new approaches. This is often the case with MBSE methodologies and tools. Simplified user interfaces, complexity management through different levels of detail, and easy navigation and immediate visualization are key aspects to consider.
- **Annoying constraints:** each company must take preventive action to avoid unwanted flow of sensitive data, propagation of human errors or negative impacts on company performance and security. This is typically translated into necessary constraints that are considered annoying limits by the end user. Tools that allow more fine-grained control of the flow of information may ease the constraints and improve overall security.
- **Demanding collaboration and different cultures:** collaboration among different users and stakeholders is typically a source of misunderstanding and requires well established processes and procedures. Even when consolidated in practice

due to past experience, any change brings new risk and should be disseminated correctly. When using different tools, this also translates into interface and compatibility issues (between both tools and people).

4 Proposed Extensions

Three main extensions to the current modelling solutions can fill these gaps:

- (1) Extending the current data models with knowledge-oriented and simulation-oriented concepts, leading to the definition of an executable virtual product.
- (2) Extending the current data models to include all statuses of a product (in PLM, from as-required to as-designed, as-analysed, as-built, and as-maintained), closing the gap between early engineering studies, detailed engineering tasks, production, testing, operations and maintenance.
- (3) Extending the data models, related tooling and current processes to derive the system definition as a set of engineering services and related connections to project and company management.

4.1 Knowledge- and Simulation-Oriented Concepts

There are two main needs in the modelling and simulation field: (1) a modelling methodology that is generic enough to not constrain the solution definition, and (2) a modelling methodology that is specific enough to allow data to be univocally interpreted.

For example, SysML responds to the first need but is lacking in the second need, so a typical user specializes and customizes the language to meet their needs. The current trends (e.g., use of ontologies, the semantic web, and more detailed data models) respond to the second need but their definition and interpretation is often limited to IT experts. A potential solution is to decouple the two needs so that (1) generic concepts (such as the fact that a product is composed of other products and may be interfaced with other products) could be defined in a generic modelling methodology, adopted across different industrial domains and types of expertise, and (2) specific concepts are standardized at a community level (a community may be related to a specific industrial domain, a specific scientific field, or to a specific project or team).

This approach can also replace the current standardization in document-based approaches. The end purpose is to produce models with items whose semantics can be understood by any target user and can be interpreted univocally by a machine (e.g., a simulation software code). This would allow a transition between a view of

the MBSE as a model-based description of a system and related systems engineering data to a view of MBSE as the definition of a virtual product that can be interpreted and executed throughout the life-cycle, with capabilities related to a specific definition level of detail, input between users and the different product variants.

4.2 From Definitions to Realizations

Current data model support in MBSE tools– and the prevalent standardization effort of SysML–lacks explicit support for including the status of a product, from as-required to as-designed, as-analysed, as-built and as-maintained. This can be an obstacle for data interchange among tools employed throughout the product life-cycle (such as PLM tools). Using the example of the mass of a system component, a component may have an as-required mass, an as-designed mass and an as-measured mass. All three must be stored and tracked throughout the system life-cycle.

A first step is differentiating between component definitions and component realizations. Each component definition can be realized many times and each component realization corresponds to only one component definition. This difference is made explicit by recent MBSE efforts (Rey 2013) promising but still focused on a specific industrial sector, biased by the specificities of one-of-a-kind-products.

For example, the design of a product can be modelled by a component definition. The product can be realized (manufactured) many times. Examples of component realizations and definitions are the manufactured units and their designs. The component definitions can have an associated as-designed mass and component realizations can have an associated as-measured mass. Design upgrades can be modelled as component definition versions and product upgrades due to maintenance can be modelled by component realization versions.

4.3 Service-Based Engineering

The conceptual infrastructure that defines the data structure of a system model has a key role in the management of the available information. A clear representation of all possible data sources and their relations is fundamental for designing an effective system. The development of modelling processes in which the customer is increasingly involved within the design activity shows promising capabilities for the near future. Customer-in-the-loop strategies highlight interesting benefits regarding the expected system performance and a better exploitation of available resources. A clear and deeper involvement of the customer in the decision-making process can help generate a product that is better aligned with market expectations.

This aspect is common to different markets and the same approach can be used, with minor changes, across different domains.

These considerations highlight how a conceptual infrastructure for service modelling can help include additional scenarios in the context of system definition. The development of the objects and the relationships that characterize a service can enhance the communication between customers and system designers. This vision of the customer-in-the-loop strategy can be pursued through application of a model-based philosophy, providing all the features and benefits that aid in the definition of a system project. The related data structure can be used to drive information exchange between stakeholders and track of the current baseline and changes in a consistent manner.

5 Conclusion

The experiments conducted in UIW, especially those in the space and ship building domains, show interesting results for the extension of system models. Extensions include web-based collaboration, the connection with simulation and virtual reality, and the use of services and probes. However, the experience with causal context models, the circular economy model and other strategic/company level models should be linked with technical choices, collaboration aspects, project management and company strategy. Models that relate economic and strategic domains are often difficult to formalize, but a clear integration of the related concepts with the current system model can greatly enhance the lifecycle process.

Extending the model to include maintenance activities yielded interesting and promising results. Additional details are presented in Chapter “[Collaborative Management of Inspection Results in Power plant Turbines](#)” in the context of maintenance of power plant turbines. Efforts for harmonizing the product structure from the assembly point of view with alternate methods of structuring information derived from the maintenance processes have shown improvements in communication and information sharing among the different actors involved. However, extending the system model to cover both the product and the service views would require changes to the companies’ information systems. However, such a step and the adoption of an MBSE approach, which is not the case for some traditional industries, are required to ensure the convergence of the actual practice towards a more effective data management solution.

The UIW-experience allowed determination of the common issues among different design processes. Such problems can be faced with a more effective design approach, and a model-based philosophy can provide useful tools to mitigate the current situation. Additional features can be integrated within the system model to cover common areas among different companies. For example, a well-formalized system model can pave the way for tools and techniques that can support the decision-making process. Currently, system solutions and design choices are strictly dependent on the context and seldom can all the knowledge elaborated during these processes be re-used in

other projects. It is often difficult to track the rationales that drive a design choice because it is difficult to formalize how such information can be defined. An extension of the system model to include such aspects can improve the design process across different industry domains, especially with respect to company strategies and objectives. The system data can be exploited in a more effective manner if defined properly following the pattern of a formalized system model. In this manner, the information collected in a project can be re-used in another one with less effort than the traditional approaches. Company expertise can also be managed in a more consistent manner to help correlate all available information for product design.

For example, the decision-making process can take advantage of a model-based approach because optimizations, trade-offs or sensitivity analyses can be performed consistently with the available system data. Another interesting advantage is that all the related information for optimization, trade-off or sensitivity analyses is not only helpful for the current design but can be re-used in the future because it follows the structured data representation of a common system model. However, such an extension of the system model requires a clear understanding of the current optimization or sensitivity analyses practices so that the largest number of design scenarios is covered. The variety of optimization or sensitivity analyses scenarios is generally broad due to the characteristics of the computational models, inputs and outputs.

References

- Blondelle, G., Bordeleau, F., & Exertier, D. (2015). Polarsys: A new collaborative ecosystem for open source solutions for systems engineering driven by major industry players. *INSIGHT*, 18(2), 35–38.
- Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5), 61–72.
- Cencetti, M. (2014). *Evolution of model-based system engineering methodologies for the design of space systems in the advanced stages of the project (Phases BC)*. Politecnico di Torino. Retrieved from <http://porto.polito.it/2572760>
- de Koning, H. P., Gerené, S., Ferreira, I., Pickering, A., Beyer, F., & Vennekens, J. (2014). Open concurrent design tool—ESA community open source ready to go! In *Presented at the 6th International Conference on Systems and Concurrent Engineering for Space Applications, Stuttgart*. Retrieved from <http://congrexprojects.com/2014-events/14c08/>
- Di Giorgio, F., & Wiart, B. (2012). Applying collaborative system engineering in Thales: Lessons learned and best practices. In *Presented at the 5th International Workshop on Systems & Concurrent Engineering for Space Applications (SECESA 2012), Lisbon*. Retrieved from <http://www.congrexprojects.com/12c12/>
- Forsberg, K., & Mooz, H. (1992). The relationship of systems engineering to the project cycle. *Engineering Management Journal*, 4(3), 36–43.
- Friedenthal, S., Moore, A., & Steiner, R. (2014). *A Practical guide to SysML, third edition: The systems modeling language* (3 ed.). Morgan Kaufmann.
- Intercax. (2016). *Syndeia (formerly SLIM for Systems Lifecycle Management)*. Retrieved from intercax.com

- Karban, R., Weikiens, T., Hauber, R., Zamparelli, M., Diekmann, R., & Hein, A. (2011). *CookBook for MBSE with SysML*. SE2 Challenge Team. Retrieved from <http://mbse.gfse.de/>
- Object Management Group. (2013). *Systems modelling language (OMG SysML) V1.4*. Retrieved from <http://www.omg.org/spec/SysML/1.4/>
- Object Management Group. (2015, June). *Unified modeling language (UML) V2.5*. Retrieved from <http://www.omg.org/spec/UML/2.5/>
- Pasquinelli, M., Basso, V., Mazzini, S., Baracchi, L., Puri, S., Gerbaz, D., ... Viitaniemi, J. (2014). Model-based approach for the verification enhancement across the lifecycle of a space system. In *6th International Conference on Systems and Concurrent Engineering for Space Applications*. Stuttgart. Retrieved from <http://ceur-ws.org/Vol-1300/ID11.pdf>
- Rey, J. (2013). *Modeling with VSEE: Definition of guidelines and exploitation of the models* (p. 48). European Space Agency. Retrieved from <http://www.vsd-project.org/download/documents/YGT%20final%20report%20Rey%20V2.pdf>
- Roques, P. (2016). MBSE with the ARCADIA method and the Capella tool. In *8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)*, Toulouse, France. Retrieved from <https://hal.archives-ouvertes.fr/hal-01258014>
- Rouhani, B. D., Mahrin, M. N., Nikpay, F., Ahmad, R. B., & Nikfard, P. (2015). A systematic literature review on enterprise architecture implementation methodologies. *Information and Software Technology*, 62, 1–20. <https://doi.org/10.1016/j.infsof.2015.01.012>
- Royce, W. W. (1970). Managing the development of large software systems. In *proceedings of IEEE WESCON* (Vol. 26, pp. 1–9). Los Angeles. Retrieved from http://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf
- Spangelo, S. C., Kaslow, D., Delp, C., Cole, B., Anderson, L., Fosse, E., ... Cutler, J. (2012). Applying model based systems engineering (MBSE) to a standard CubeSat. In *Aerospace Conference, 2012 IEEE* (pp. 1–20). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6187339
- Technical Operations International Council on Systems Engineering (INCOSE). (2007). *INCOSE systems engineering vision 2020* (No. INCOS E-TP-2004-004-02). International Council on Systems Engineering. Retrieved from http://oldsite.incose.org/ProductsPubs/pdf/SEVision2020_20071003_v2_03.pdf
- Wiley, & Others. (2015). *INCOSE systems engineering handbook: A guide for system life cycle processes and activities*. Wiley. Retrieved from <https://books.google.es/books?hl=es&lr=&id=8-cbBgAAQBAJ&oi=fnd&pg=PA145&dq=Systems+Engineering+Handbook+INCOSE&ots=wjK2AWI7Nj&sig=auPfFAItomJSjQ0axlwqK4IHm58>

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

