INCOSE Italian Chapter Conference on Systems Engineering (CIISE2014)
Rome, Italy, November 24 – 25, 2014

Model-based approach for the verification enhancement across the lifecycle of a space system

M. Pasquinelli, V. Basso
Thales Alenia Space Italia,
strada Antica di Collegno, 253
10146 Torino, Italy
Email: {mauro.pasquinelli,
valter.basso}
@thalesaleniaspace.com

D.Gerbaz, J. Fuchs
European Space Agency
Keplerlaan 1
2201 Z Noodwijk, the
Netherlands
Email: {diego.gerbaz,
joachim.fuchs}@esa.int

S.Mazzini, L. Baracchi, S. Puri
INTECS
via Umberto Forti, 5
56121 Ospedaletto - Pisa, Italy
Email: {silvia.mazzini,
laura.baracchi,
stefano.puri}@intecs.it

L.Pace, M. Lassalle
Politecnico di Torino
corso Duca degli Abruzzi, 24
10129 Torino, Italy
Email: {lorenzo.pace,
marco.lassalle}@polito.it

J. Viitaniemi
Tekniikankatu 1
33101 Tampere, Finland
email: juhani.viitaniemi@vtt.fi

Abstract. This paper reports results of the ESA MARVELS study (Model-based approach research for the verification enhancement across the lifecycle of a space system), with the objectives to define adequate model-based methods to improve the overall verification process of space systems, and to define, prototype and integrate supporting tools for System Verification along the entire project life-cycle.

The study results aim to demonstrate how it is possible, in the near term, to perform a transition to a full Model Based System Engineering (MBSE) approach for the verification process, including re-use of elements from past projects, and definition of new ways to support a more effective review process. The feasibility of shifting to this innovative approach in the short term is proven by the definition, prototyping and demonstration of a suitable methodology and environment to support the activities of verification managers and practitioners from different industrial levels, along the lifecycle.

Introduction

The space sector has consolidated ways to control a large project, guiding the project teams from the system definition up to the final product realization and usage, challenging the technical, industrial and operational complexity and keeping the mission success as a main driver. One main key to the effectiveness of the approach is the clear and correct definition of the specification and related verification process, which guides technically and contractually all the project stakeholders.

According to the success of most of the recent European missions, the current verification process seems to be effective[1], i.e. it is able to anticipate on-ground problems that could have

caused severe consequences if detected in orbit. The related processes are well established [2] and we can sustain that the practice has currently reached a good degree of optimization, with respect to the current methods and, of course, excluding typical inefficiencies, intrinsic of the organizational complexity, which shall be taken into account.

The current process is document-based and, as defined in the ECSS [2], the main products of the verification are documents and reports. The origin of the study was the question: "what can happen for system verification if we move from a document-based approach to a model-based approach?"

Such question drives the work of the MARVELS study team, composed by Thales Alenia Space Italia (prime, MBSE and Verification expert), INTECS (MBSE methods and tools expert, prototype developer), Politecnico di Torino (calculation tools and space systems engineering expert) and VTT (socio-technical systems and industrial processes expert).

This paper provides an overview of the results, supported by examples, showing how a model-based approach can impact the following process steps:

- System requirements definition and formalization
- Validation/Verification definition
- Verification activities (e.g. Test/Analysis) Specification
- Test/Analysis Models generation and configuration
- Test/Analysis Results management and processing
- Verification results, reports and status management/Verification Control

For each of these process steps, with special focus to the system requirements definition and formalization, the paper shows the application of the investigated theory and the main expected improvements through the adoption of the model-based approach, as well as the available or needed enabling methods/technologies/tools. This is done through:

- The modelling of the spacecraft system description
- The formalization of requirements into constraints and their application to the associated model items through an innovative contract-based approach, formalizing assumptions and constraints, so that they can be re-used in the future as checks against the satisfaction of some constraints
- The constraints check directly on the model (applicable for specific classes of constraints)
- The modelling of the selected verification strategy
- The connection between the system model and related analysis or test models
- The post-processing of analysis or test results supported by formalized constraints (where possible) and the collection of such results connected to the associated modeled verification activities
- The control of the verification status and the final generation of the Verification Control Document.

Such approach is facilitated thanks to a demonstration scenario based on real cases, inspired by or taken from two very different programs: the new water pump assembly for the International Space Station (ISS) ESA Columbus module Water Pump Assembly (WPA Mk II) and the Exomars 2018 mission. Suitable pieces of the related systems information are modelled, associated with formalized requirements and verification information, linked with analysis and other verification activities, whose results are then back propagated to the model itself and used to provide the current status of the verification activity, with the final purpose of managing and closing the Verification Control Document (VCD).

An overall supporting tool architecture is also described as required by functional and modelling needs, showing also what is currently available at research/prototype or operational level. The next steps in the study are presented at the end.

Next sections provide an overview of the study output, starting with the type of models that we considered, how they are connected with validation and verification activities in the project lifecycle, how such activities may be supported from the early stages of the process by a formalized system model, and which are the new potential activities that are enabled by its usage. A brief description of expected benefits and next steps concludes the paper.

Overview of the Model-Based approach in the system life-cycle

An MBSE model can be defined as a formal representation of an abstraction of the system of interest w.r.t. specific process objectives. In MARVELS, the system of interest is a collection of entities/actors (operational entities, project actors), products and processes/services (simulations, model exchange, etc.), and related interconnections, related to the space segment systems that are designed, produced, delivered and supported during operations. The environment, intended as the existing surroundings of a product that may interact with the system by exchanging information, mass, energy, or other properties, is considered included in the entities/actors.

The transition from a document to a model means that the related information is formalized in terms of specific items, whose semantics can be understood by a person, but also processed by a computed algorithm. A simple example can be the difference between the sentence "the mass of the equipment is 10 kg" and the value "10", associated with the unit "kg", associated to the product "equipment", where *value*, *unit* and *product* are meaningful and unique concepts, which are defined and standardized in the operational context where they are used.

Many types of models can be used in a model-based environment, and some of them are already used to a large extent, especially in some engineering disciplines. A classification limited to engineering activities may be:

- System-level models, can be abstracted as:
    - System/Architectural descriptive models: they are used to describe system-level data in a formal way. Examples are the OMG UML [3] (mainly for software) and SysML [4] (for systems), the ESA Open Concurrent Design Tool (OCDT) [5] (for preliminary design) or the ESA Virtual Spacecraft Engineering Environment (VSEE) [6] (intended for the whole lifecycle), the Thales Melody Advance model [7] or the Thales Alenia Space Italia Distributed Environment for Virtual Integrated (DEVICE) model [8]
    - CAD models: they are used to define, control and maintain the physical configuration, the items arrangements, the related interfaces and harness routing. CAD models are currently supported by many commercial tools.
- Engineering discipline models, can be abstracted as e.g.:
    - Geometrical models: they represent a derivation of the system geometry (defined in the CAD) for discipline specific calculations and control. Examples are meshed surfaces or volumes for FEM/CFD analysis or simplified shells for lumped-parameters analysis.
    - Descriptive models: they represent a simplification of the system architecture from a specific discipline point of view
    - Calculation models: they represent a simplification or a discretization of the system of interest, in order to be able to solve problems represented by partial differential equations (PDE's) for Analysis/Simulation of continuous models, or differential algebraic equations (DAE) for Analysis/Simulation of discrete models
    - Logic/SW models: they represent specific logics to be implemented in the system software, controllers or simulating the behaviour of external operational entities.

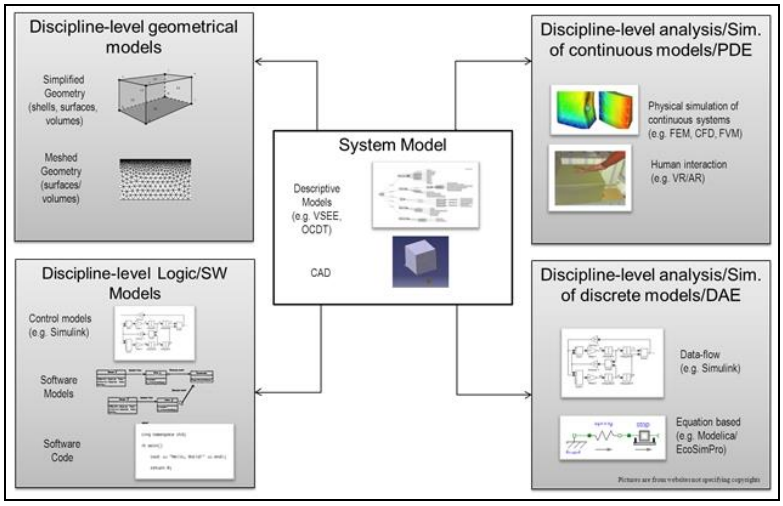

Fig. 1. System and discipline-specific models

Of course many of these models have overlapping pieces of information, which shall be kept consistent. According to the INCOSE vision [9], a common MBSE reference model (called System Model) is the basis for ensuring consistency, and allows each type of model to be in line with a common architecture, as shown in Fig.1. This approach can provide a substantial improvement to the design activities, though this was not the focus of the study. The study concentrated on the considerable improvement of the efficiency in some of the verification activities, as described in the next sections, together with some exemplary cases.

Verification definition and management supported by modelling

Fig. 2 provides an overview of the main differences between a document-based verification process and a model-based one. The documents that are produced in the traditional process, reported in (a), could be replaced by a system model which is able to include system requirements, the specification of validation and verification, and of their activities (e.g. test or analysis), which is linked with test and analysis models (including flight units), related results and reports, as outlined in (b). If documents are still required, they could be generated from these models. This, associated with the possibility to interface customer and suppliers providing the acceptance of the verification status by the customer, provides a good means for verification control and for the generation of the related document (VCD) when needed.

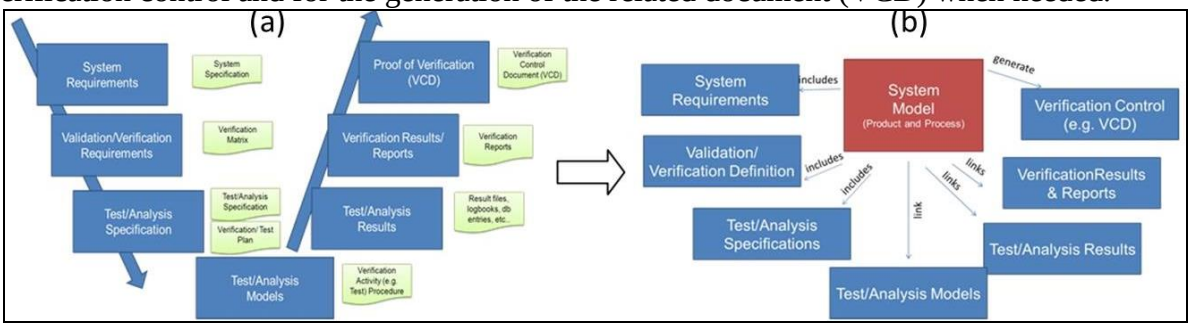

Fig. 2. Verification process: from document-based to model-based

Next sections show potential impact of the model-based transition on each of the related activities, i.e. on:

- System requirements definition: through formalization and re-use
- Validation/Verification definition: through appropriate links with verification methods, stages, levels and models
- Verification activities (e.g. Test/Analysis) Specification: through appropriate links with requirements and architecture

- Test/Analysis Models generation and configuration: through different levels of support to the related practitioners
- Test/Analysis Results management and processing: helping the reporting and control activities
- Verification results, reports and status management/Verification Control: helping the verification manager to control and check the status

Verification and the importance of requirements

A good verification work relies on a solid requirements specification. Current requirements management methodologies already have been improved since the traditional monolithic documents, through the use of centralized databases with traceability functions. A model-based approach enables a substantial improvement to such enhancements, thanks to the possibility to express each single requirement not only with associated traceability with the related engineering item (e.g. an architectural element, an operational activity or a specific value of a property), but also expressing the requirement itself as a model, expressing the related constraints in a formal way.

Our interest here is to support the system engineer to build a model that complements the text requirement by expressing both *problem oriented* and *solution oriented* statements. Considering the example provided before, the interest is to model the requirement "the mass of the equipment is 10 kg" satisfied by the product "equipment", that has the attribute "mass" with value "10", associated with the unit "kg", in the system model.

A requirement expresses constraints on the product (element) to be developed or to be verified, or on the related processes. There can be many types of constraints, a majority of which can be generalized in two classes:

- Value constraints, i.e. constraints on acceptable ranges or imposed values w.r.t. the property of an item defined during the engineering activity
  - Example: element property (the mass shall be less than * kg), range of properties (e.g. the temperature shall be kept between the operative and non-operative ranges defined in table *), or performance (e.g. the Absolute Pointing Error (APE) shall be less than * arcsec)
  - Example: activity property (e.g. instrument calibration shall have a duration of less than * months)
- Structure constraints, i.e. constraints on the presence, absence or a number of specific types of items
  - Example: architecture constraints, imposing the presence of a certain number of items of a determined category (Two pressure sensors shall be implemented and used for ISS EWACS monitoring)
- Formalized constraints are necessary, but not sufficient to express a requirement. In fact, the limits imposed by a requirement are also dependent on a certain number of assumptions (e.g. environmental conditions, operational activities, duration or boundary conditions). Assumptions are transversal to requirements, but are necessary to establish what are the limits for the validity of the constraint, or for the validity of the actual design solution, or determine which are the conditions of the associated tests or analysis.

Typically, assumptions derive from applicable specifications (e.g. environmental or interface specification), or from design choices, and can be modelled as other requirements. Applicable specifications can be collected in similar structures as repositories of requirements.

Constraints and assumptions are then associated with the verification activities that shall verify them on a specific element (defined or realized).

Once the verification is successfully performed, the verified constraints, according to specific assumptions, are considered as guarantees, justified by the performed verification. The qualified or accepted element is therefore associated with a "contract" which is the collection of the guarantees, the assumptions and the performed verification activities (which may also be associated with the related analysis and test models and results).

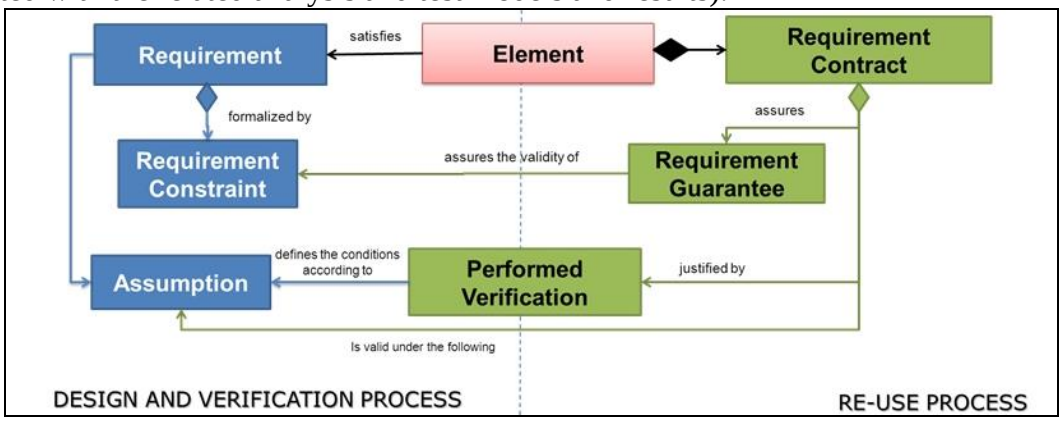

Fig. 3. Requirements, Architectural Elements and their re-use

Such contracts are the data items to be analyzed in subsequent projects, in order to control and check if the new project has specific assumptions or constraints that are inside the boundaries of the previous project, so as to avoid additional risks.

The adoption of a contract-based design (an emerging approach in research programs to address the increasing complexity of industrial systems [13]) offers advantages in terms of separation of concerns, expressiveness, modularity and re-usability.

Verification activities supported by modelling

Verification practitioners are those who are requested to provide proof of evidence that one or more requirements are met, through one or a combination of Test, Analysis, Inspection or Review of Design - the four methods prescribed by the space engineering verification standard ECSS-E-ST-02C.

For **analysis or simulation**, a *calculation model* is needed. The necessity to perform system or interdisciplinary calculations may profit by the use of a Model-based approach and the connection between such calculation models. Such models can contain either a mathematical model (formed by equations, system of equations, power series…) or an algorithm (in turn containing equations, look-up tables and other calculation devices, usually in the form of an executable script). It does not contain numerical methods that instead are part of the solver.

In order to keep consistency, *calculation* and *descriptive* model (i.e. constituting the system model) should use the same *model libraries*, containing recurrent categories, elements, value properties, items; some *model libraries* could be specific to one kind of model, as for numerical methods that are peculiar of the *calculation model*.

The descriptive model, used by the MBSE architecture and process, may help the disciplinary (and especially the multidisciplinary) analysis:

- Configuring the calculation model through a transformation from the descriptive model properties and data item, through dedicated mapping. This allows a better integration between different calculation models, originated by the same system descriptive model, and provides a further means to validate the correct definition of the system descriptive model

- Checking analysis results with the calculation model (if the function is available) or at system level in the MBSE platform, if the requirements are formally expressed, or the user has proper means to analyze them

What is called "transformation" can be provided in three different ways, with an increasing degree of complexity:

- **Subscription:** the analyst requests and subscribes to the items that impact on his/her model, and is notified each time there is a change
- **Formatted input:** the analyst prepares an input file (e.g. a text file, a MS Excel workbook, a Matlab file) in the format that can be used to configure his/her parameterized model
- **Model-to-model:** the analyst writes some code that is able to generate a discipline specific model, according to specific data items (e.g. belonging to particular categories or with specific properties)

Of course, the last point is not expected to be put in place in the near term, but it enables many new opportunities, from the design and the verification perspective.

MBSE may support **Review of Design** (RoD) thanks to a wide coverage of the description of the design through models. The model of requirements contains the formalized system specification. The descriptive model contains physical parameters, attributes and logics; it does not necessarily contain physical equations that describe the physical laws which the system is subjected to, but may contain equations and/or algorithms if their definition is an objective of design.

Review of design, when MBSE is applied, may be substantially supported by model checking. Model checking can be performed automatically if all constraints for the correctness of the model are known and formalized, otherwise it requires some degree of manual/human intervention.

MBSE may integrate separately several aspects useful for verification by **inspection**.

It may support inspection preparation, through the evaluation of requirements verifiable by inspection and the identification of hardware needed for inspection. It may be used for inspection post processing, for the evaluation of impact of corrective actions. Finally, it can lead to the virtual inspection, allowing the highlighting of criticalities and the verification of human interface aspects. This would be a borderline application, sharing some aspects with the review of design verification method.

Test preparation can be supported by MBSE for several aspects, starting from the development of test specifications and test procedures up to the identification of hardware needed for test procedure, to the validation of testing software and the set of testing parameters; this is reflected by:

- The individuation of critical conditions w.r.t. requirements
- The evaluation of representativeness of tests w.r.t. real conditions

A formalized structure also helps for post processing of data after test execution.

A better exploitation of data can be achieved through an accurate understanding of test results. Sometimes non-conformances are not spotted even if the test is correctly performed, because exploitation of test results is not adequate; detection of non-conformances on recorded parameters could be eased through the involvement of the requirement model and automated procedures for the scan of testing outputs.

Moreover, non-conformances could be assigned more correctly to the cause that originated them with the support of an adequate architecture, able to reconstruct causing events. In this way, it is easier to achieve a proper individuation of failed items. Also, it is possible to evaluate in advance the impact of corrective actions.

Not only testing activities can be supported with MBSE, but they can also give a contribution to the effectiveness of such techniques, exploiting test results for the validation of

analysis models, making them more accurate and leading to a deeper return of experience for future projects.

The aspects related to virtual testing fall inside the scope of MBSE applied to verification by analysis.

Overview of the expected impacts on project activities

A huge concern is that in companies the model based approach is not properly understood. Experiences from industry show that people are only now becoming really interested in systems approach and associated model based methodologies as they understand that these help avoiding many of the typical errors and malfunctions encountered with traditional design approach. Even if it is widely considered as a benefit, the problem is how and to what extent its implementation shall be made: the model based approach is still often perceived as a lot of cumbersome extra-work.

However some people are starting to recognize that this change in approach is a fruitful investment, a strategic decision for a company, if it is done with a proper assessment of impact on the whole sociotechnical system and on the process.

Based on valuable theories such as the systems theory approach or the activity theory [14], enforcing a holistic approach and acknowledging the concept of "socio-technical system" is fundamental: it is important to avoid that the introduction of a model-based approach is limited to technical aspects only, i.e. to the target product, overlooking the important aspects of intra-company and extra-company (social) interactions. Such interactions enable an actual co-engineering approach, and facilitate the customer-supplier relationships, maintaining a holistic approach that takes into account all the stakeholders of the project, from the designer, to the manager, to the verification practitioner, to the customer, to the user.

The intra-company collaboration is facilitated by:

- The involvement of the team members, clarity of responsibility, shared status of the project and activities, reduction of the effort spent for reporting activities (e.g. generating documentation from models)
- Common and easy-to-use tools, possibly based on common shared concepts, able to give the team members the capability to produce, share and consume information in an easy way

Producing, sharing and consuming information in an easy way requires the sharing of a model. As shown in Fig. 4, for the space domain, sharing a model in the different phases of a project requires at least three main blocks: sharing the underlying concepts (e.g. what is a function, a product or a value), sharing specific libraries (e.g. what is and how to call the mass property, what is and how to call a battery), and the sharing of the project specific data items (i.e. the project baseline items that should be shared inside the team or between teams).

In the verification process this is deemed important, because a mismatch between the meaning of the concepts in a team and their meaning in another team may invalidate the related efforts. This issue is already experienced in the document-based approach, and it is one of the reasons for big standardization investments that have led for instance in the space domain to the definition of the ECSS standards and handbooks, which are then tailored for each project, are applied in different ways in each company reference processes and then customized by each project team. The model-based approach should provide similar ways to cope with this complexity, but formalizing shared concepts in object libraries.

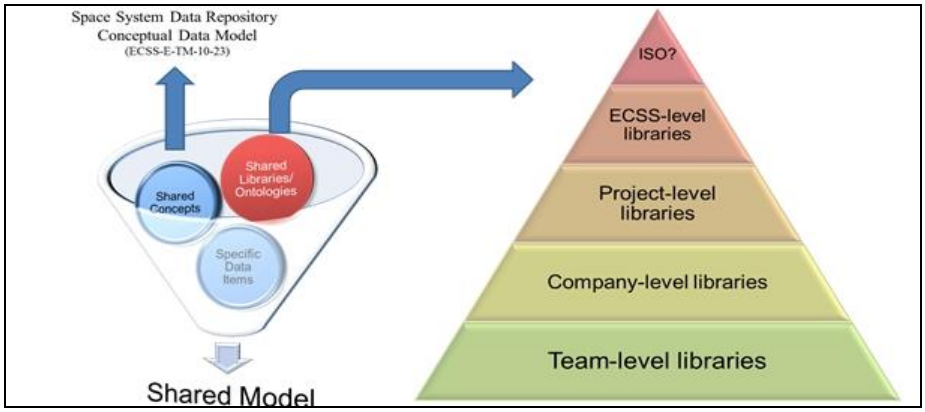

Fig. 4. Sharing data at different levels

In the customer-supplier relationship one of the key improvements is expected in the review process, providing means to share continuously the status of verification activities, to be able to close and archive agreed items in a lighter way and to share prioritization of issues and other items to be reviewed. In particular the following changes are expected in the review process:

- Reduction of RIDs
  - Customer-Supplier strengthened co-engineering should reduce RIDs and identify key issues
  - Other key issues are identified by a third party team (review team)
- Co-engineering
  - Customer-Supplier as a team
- RIDs are identified directly on the model
  - In early phases this shall improve the quality of the model
- The data-package is generated for records once the RIDs are solved, and for the review authority, not for the whole review process
  - Reduced time for documentation generation
- The intervention of the review team can be associated not only with a review stage, but during the phases, once the items are stabilized
  - Third party assessments can be easily requested by customer or supplier on specific tasks, because they do not require documentation

An example of potential benefits is depicted in Fig. 5, where the required effort for the review is estimated, providing a better concurrent engineering since the project phase activity, reducing the review effort, but keeping the objectives and effectiveness.

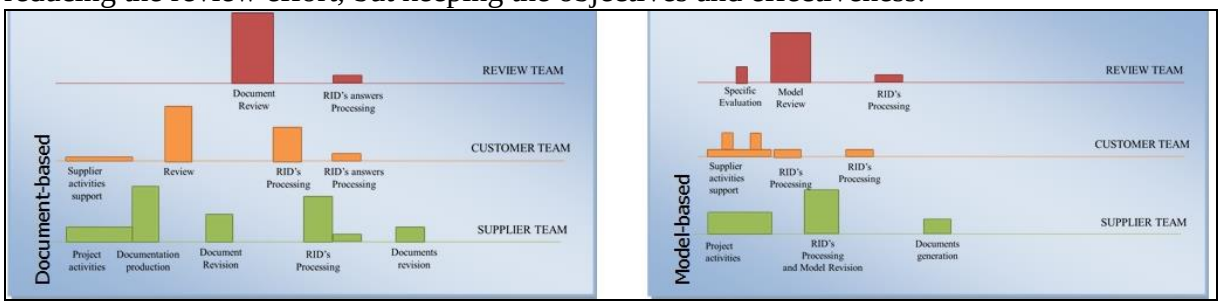

Fig. 5. Supposed effort required for review process

A contract based approach and supporting toolset

A modelling framework was developed by INTECS, based on the EMF Eclipse-based model editor and SysML, to provide a demonstrator for the presented methodology, allowing to experiment modelling with the tool, integrating VSEE models, and to evaluate the potential

enhancements to the verification activities along the development life-cycle, in particular in the early development phases.

The demonstrator focused on the contract-based approach in order to highlight its advantages in terms of:

- co-engineering - the adoption of contracts at all stages of system design, from early requirements capture, to any level of system modelling and detailed design involving software and hardware, is particularly convenient as a means to ease communication between customer and supplier and also between different teams within a same company in charge of the development of different subsystems or different aspects of a system
- enforcement of separation of concerns - it is a key distinguishing feature supported by the use of contracts (even if simply expressed as textual requirements in an informal language): what must be guaranteed by the system itself is kept separate from what are the constraints on the environment required for the system to enforce its guarantees
- component reuse - contracts can be considered formalizations of the conditions for correctness of element integration, for lower levels of abstraction to be consistent with the higher ones, and for abstractions of available components to be reused
- systematic virtual integration and verification - assuming that the virtual integration verification was successful, it is possible to verify whether the system itself meets its contract purely based on the knowledge of the subsystems contracts and the system's architecture (and evidence that the subsystem implementation is compliant with this contract)
- protection of intellectual property - the contract-based approach to virtual integration verification allows the intellectual property of subsystem suppliers to be protected as the only evidence required is the confirmation that their implementation meets the subsystem contract specification.

Particular emphasis was dedicated to supporting the definition, formalization and modelling of requirements and contracts according to the presented methodology, and to supporting the automatic verification of constraints on the model. Other tooling features useful to support the process were outlined, but not fully implemented due to the tight project's timeframe.

Inclusion of requirements in the system model as first class citizens, strongly interconnected with the other model elements according to a SysML-like approach, is mandatory for the definition of a development methodology where significant verification activities are carried out from the early stages of the process.

Requirements evolve and are enriched across the whole life cycle, with progressive refinement taking place in parallel with the hierarchical decomposition of the system's architecture following an iterative and incremental process.

Formalization of requirements plays a central role in the process. The formal definition of a requirement (*formal requirement* in the following) was therefore added as an optional field to the Requirement entity in the meta-model.

Formal requirements are structured according to the contract based paradigm, defining the assumptions under which the requirement should hold, and the guarantees that constitute the requirements' actual desiderata or constraints.

The MARVELS framework provides editor-assistance to support:

- the creation of formal requirements matching a predefined grammar
- linking of constraints to model elements and properties
- automatic verification of formal requirements

The MARVELS framework (sketched in Fig 6) provides a model editor with assistance for the creation of syntactically correct formal requirements by matching a pre-defined grammar

based templates for the identified classes of formal requirements (i.e. value constraints and structure constraints).

For the sake of generality and in order to maintain the highest possible level of independence from the VSEE meta-model [9], the constraint templates were implemented to be agnostic w.r.t. the underlying meta-model. They needed meta-information is dynamically retrieved from the model (e.g. containment relationships for structure constraints).

Semantic correctness of constraints is guaranteed by the editor's awareness of the model structure and element's properties: where necessary, the user is prompted to choose among existing model elements/attributes with correct scoping: the editor dynamically extracts info from the model for the composition of the formal requirement.

Syntactic and semantic correctness is however not strictly enforced by the tool: creation of formal requirements with references to non-existing model elements and/or attributes – though highlighted as incorrect - is permitted in order to allow the definition of formal requirements on system components even before the corresponding level of decomposition has been modelled.

The opportunity to perform early verification on virtual models is further enhanced by the automatic verification of requirements on the model. This is achieved by comparing the constraints in the formal requirements with the corresponding model elements and properties they are linked to.

For structure constraints the verification can be performed directly on the model's structure. For value constraints, instead, only in some cases the check can be carried out directly on the model (if the model element that is the subject of the requirement has a value associated to the constrained property). In general, the calculus for value constraints is performed by specific engineering discipline calculation, analysis, simulation, or by testing. In many cases the check may be carried out on result values imported in the model. For example, the result file produced by a discipline calculation tool may be automatically retrieved and processed based on its known format and the constrained value can be evaluated.

When a formal requirement is successfully verified on the model, the requirement's status is automatically updated and the SysML *Satisfy* relationship is created in the model to link the requirement with the element that satisfies it.

The corresponding contract, derived from the formal requirement, is also automatically created in the model and linked to the model element that satisfies it, as a label with "instructions" for future use.

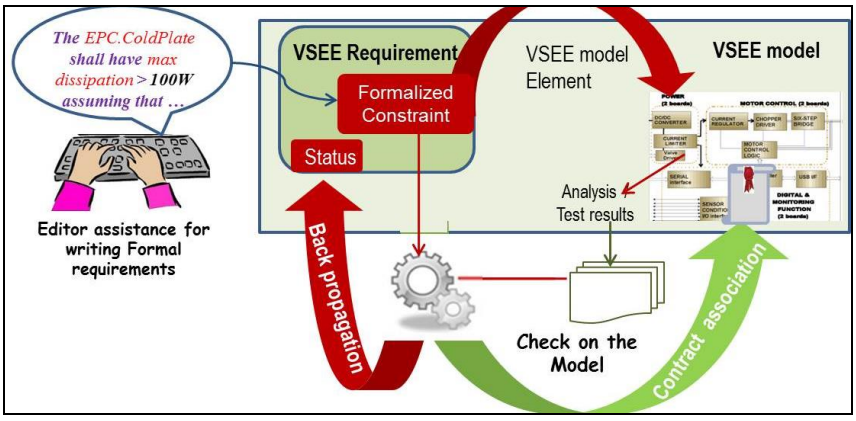

Fig. 6. The MARVELS supporting framework – an overview

Demonstration

The study conclusion is scheduled by the end of 2014. The last phase of the study is devoted to the demonstration of the explained concepts using as demonstration cases the WPA Mk II (a small, but complex system), and the Exomars 2018 mission. Some examples are used to

provide appropriate links between the system model and analysis models that can be used for (or in support to) verification.

A whole demonstration is developed, using an existing model-based environment (based on VSEE), providing support for requirements and contract formalization (thanks to INTECS developments), linking the model to calculation tools and test results (thanks to POLITO support) and providing an analysis of the user experience (thanks to VTT involvement).

This enables the evaluation of the feasibility of the introduction of the studied methodologies in a pilot project, or of the necessary delta's that should be defined or developed in order to let this approach become feasible, sustainable and start bringing benefits in the short term.

Conclusions and next steps

Most of the analyzed verification activities can be already supported by the MARVELS development framework, in a VSEE-like environment, and the main limit to its application is the definition of a shared process and the existence of a user-friendly, stable and acknowledged tool able to support it.

Future work may comprise further integration of the supporting tool set and the implementation of an extended version of the development framework supporting the creation of additional classes of formal requirements, and capable of handling more complex verification scenarios, for instance to deal with constrained properties that are not available per-se in the model, but can be calculated based on the model's properties. Many approaches are available in the literature to provide a solution to the question of how to formalize requirements so that they can be processed and evaluated automatically by a computer [11] [12]. A promising approach could be to support the elaboration of more complex constraints using Modelica or ModelicaML tools.

Thorough experimentation of the proposed process and toolset in industrial and research projects will be an important driver to set the path of future work, suggesting enhancements to the supporting toolset and tuning of the methodology.

References

[1] Messidoro, P., Pasquinelli, M., Pace, L., Vergès, F., Laine, B., Candè, J. 2013. *ASSET – Analysis of Spacecraft Qualification Sequence and Environmental Testing*. FADAT, June 2013, ESA ESTEC.

[2] ECSS-E-ST-10-02C, Verification, ECSS Secretariat, published 6 March 2009.

[3] Unified Modeling Language, resource page: http://www.uml.org/ - accessed 10 September 2014.

[4] Systems Modeling Language, resource page: http://www.omgsysml.org/ - accessed 10 September 2014.

[5] Open Concurrent Design Tool, resource page: https://ocdt.esa.int

[6] Eisenmann, H., Fuchs, J., de Wilde, D., Basso, V. 2012. *ESA Virtual Spacecraft Design*, System Engineering and Concurrent Engineering for Space Applications, SECESA 2012, 17-19 October 2012.

[7] Arnaudy, D., Cheutin, S. 1012. *Use of a 'Model-based' approach for EGNOS v3 System Definition*, System Engineering and Concurrent Engineering for Space Applications, SECESA 2012, 17-19 October 2012

[8] Pasquinelli, M., Basso, V., Marello, M., Paccagnini, C.M. 2012. *STEPS Results: MBSE prototypes to support Space Exploration Projects Lifecycle*, System Engineering and Concurrent Engineering for Space Applications, SECESA 2012, 17-19 October 2012.

[9] VSEE Data Model version 16, available at http://vsd-project.org/jspwiki , accessed 10 September 2014

[10] INCOSE *System Engineering Vision 2020*, INCOSE-TO-2004-004-02, version 2.03, September 2007
[11] Schamai, W., Helle, W., Fritzson, P., Paredis, C. J. J. 2010. *Virtual Verification of System Designs against System Requirements*. MoDELS Workshops 2010: 75-89
[12] Tundis, A., Rogovchenko-Buffoni, L., Fritzson, P. Garro, A. 2013. *Modeling System Requirements in Modelica: Definition and Comparison of Candidate Approaches*. Proceedings of the 5th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools (EOOLT). University of Nottingham, UK, 19 April, 2013.
[13] Sangiovanni-Vincentelli, A., Damm W., Passerone R. 2012. *Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems*. In European Journal of Control 18(3):217-238, 2012.
[14] Moser, H. A. 2014. *Systems Engineering, Systems Thinking, and Learning - A Case Study in Space Industry*. Series: Understanding Complex Systems, Springer International Publishing 2014, 328 p. 94