METHOD & TOOLS TO SECURE AND SUPPORT COLLABORATIVE ARCHITECTING OF CONSTRAINED SYSTEMS

Jean-Luc Voirin
Thales Aerospace

Keywords: *architecture modelling method early validation*

Abstract

ARCADIA is a system & software architecture engineering method, based on architecture-centric and model-driven engineering activities. It targets systems whose architecture is largely constrained by issues such as performance, safety, security. This paper emphasizes Arcadia benefits on securing system definition and verification, supporting collaborative work on architecture, and technical relationships with both customers and suppliers, along with supporting tools issues.

1 Introduction

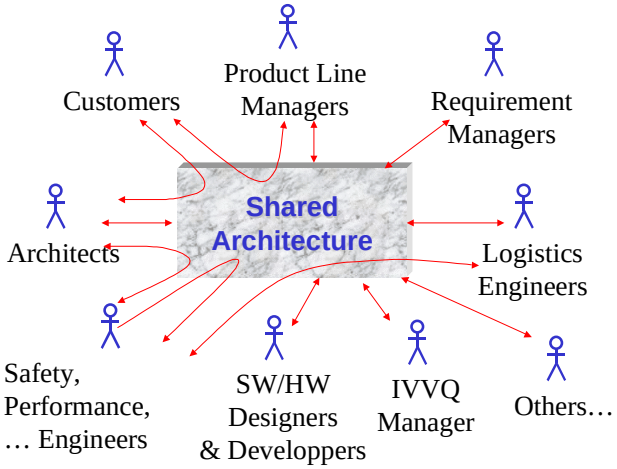
System Engineering of aerospace electronic devices and systems (e.g. avionics, flight or aircraft systems control, mission computers…) is submitted to high constraints regarding safety, security, performance, environment, human factors and more; all of these are under responsibility of different stakeholders, yet deeply influence systems architecture design and development, and are to be reconciled in a relevant system architecture.

The approach described in this paper, named ARCADIA (ARChitecture Analysis & Design Integrated Approach), based on ***architecture-centric and model-driven engineering*** activities, aims at

- securing system definition and verification,
- supporting collaborative work on architecture,
- and easing fluid technical relationships with both customers and suppliers.

Some technical features of Arcadia implementation have already been presented (see [1], but this paper focuses on and details more collaborative engineering issues.

Fig. 1. Support for collaboration in architecture building



2 Strengthening Product definition through Need analysis

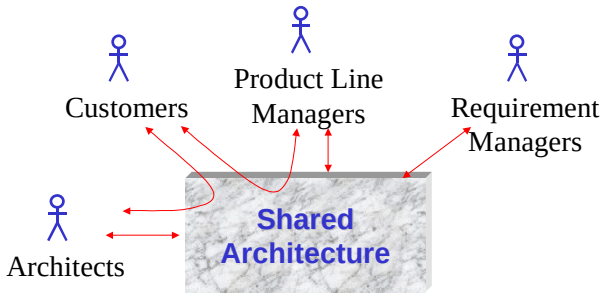
The first contribution of Arcadia to strengthening product definition comes from **formalization of need** and strong **linking with product definition**.

2.1 Need Analysis

All stakeholders concerned by system need definition are supposed to be involved in the

following need definition process, while sharing formalized need description.

Fig. 2. Collaboration in Need Analysis



Formalizing Requirements through functional and non-functional analysis

Requirements are captured and validated, not only in a text-oriented database as usually, but also formalized in a functional (e.g. see [3]) and non-functional need analysis model:

- each textual requirement, is interpreted so as to identify functions expected from the system, along with data or information to be used/elaborated, and related exchanges between functions
  Example: the requirement "the collision avoidance system shall sort and display traffic according to altitude difference" will lead to defining functions such as 'detect traffic', 'compute traffic altitude', 'compute altitude difference with aircraft', 'sort traffic by altitude difference', 'display traffic in risk of collision'. Data such as 'detected aircraft in traffic', 'aircraft altitude', will be identified, and be subject to exchanges between former functions.
- Non-functional constraints are identified and allocated to the functional analysis
  Example: the requirement "latency between traffic detection and alert display shall not exceed xxx milliseconds" will be allocated to a 'functional chain' covering functions identified above. Safety constraints could also be added as needed (e.g. criticity level of each functional chain, feared events…)
- Traceability and justification links are maintained between requirements and functional analysis

Analyzing end user needs and operational use of the system

Beside requirements, a complementary analysis is driven on operational needs and formalized in an operational analysis model:

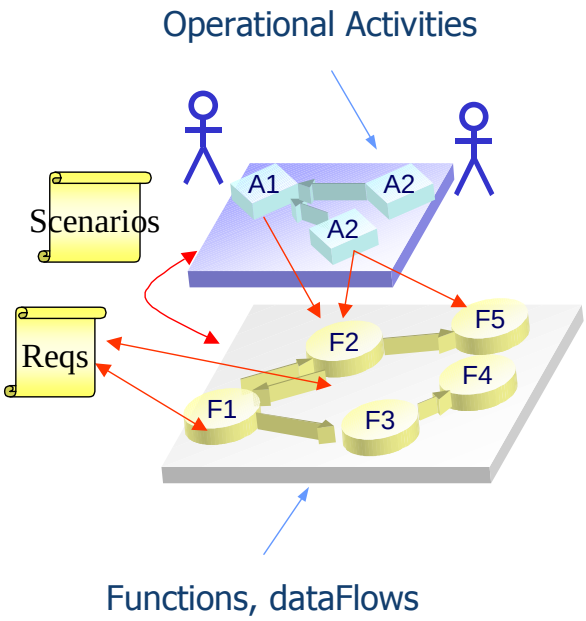
- End users missions, activities, real-life situations and operational scenarios are defined in an operational analysis model; other stakeholders involved in system exploitation are also identified
  Example of users: pilot crew, but also air traffic management, airline operations, maintenance teams…
  Example of activities: 'localize aircraft', 'monitor traffic', 'monitor primary parameters'…
  Example of operational scenario or process: in case of imminent risk of collision, the pilot changes aircraft altitude according to a predefined procedure, shared with other aircraft
- Here again, non-functional constraints are allocated to model elements
  Example: the feared event "both aircraft climb or descend" is identified and tagged as 'catastrophic'
- Starting from this operational analysis, some complementary functions can be added to the former functional analysis, based on operational activities and scenarios; they are allocated either to system or users, and linked to the operational analysis.

Checking Need validity and consistency

All three former need expression models (operational, functional, requirements) are related and confronted to each others, so as to

check consistency and completeness of need analysis, using model analysis techniques & tools.

Fig. 3. confronting Need models



Note that this approach and associated models are an efficient means to support collaboration between end users, customers, and system provider teams, so that negotiation can be more explicitly supported, and consequences of need definition easily analyzed.

2.2 Architecture building and verification

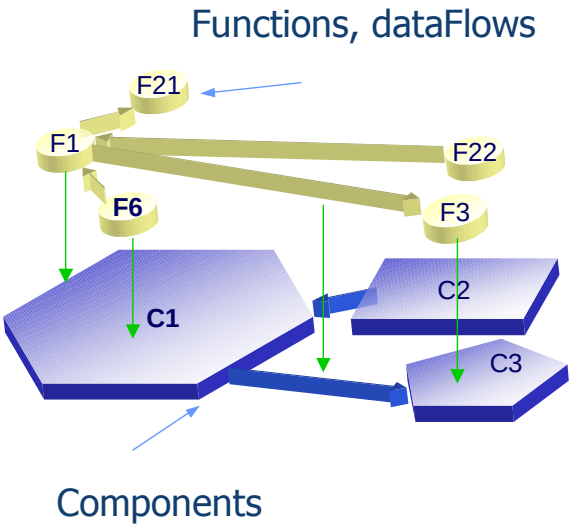
Functional analysis driving architecture definition

Product architecture breakdown is built and justified through allocation of functional need items and scenarios, on architecture components:

- requested system functions are allocated to components, either by grouping for consistency (*e.g. functional, operational, product line, interface considerations*), or by segregating them into different components (*e.g. separating functions with different safety or security levels, performance issues*).
- Interfaces and exchanges between components are derived from functional exchanges, therefore are justifiable and checkable against need.

This component definition is also driven by multi-specialty engineering considerations. Please refer to 'Non functional constraints management' later in this document.

Fig. 4. Function to Component allocation



Confronting Architecture & Design to Need analysis

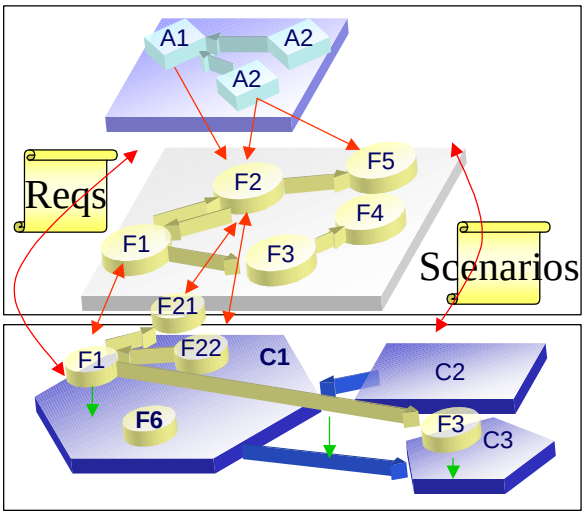
Once product architecture is defined and modeled, it is confronted to all three need models in order to check its compliance to all of them:

- Requirements are automatically allocated to components, thanks to relationships with functions of these components; if necessary, these requirements may be refined to adapt to component outline.
- component definition is built from functional analysis, therefore correct by construction with regards to this functional model
- operational scenarios are allocated to each component thanks to operational model / functional model justification links; this allows

checking component definition and allocating scenarios to component boundaries.

- this material is also the basis for further integration and validation activities and checks (e.g. reference for expected behavior & properties).

Fig. 5. justifying architecture Vs Need



This quick view of major ARCADIA steps illustrates its first means to support collaboration between need analysts, operational users, architects and designers. Let us go further now, inside engineering activities and collaborations, for other aspects of this support.

3 Non functional constraints management

The second benefit of Arcadia is related to explicit **formalization of non functional constraints**, and associated **early verification of the architecture against these constraints**.

3.1 Dealing with non functional concerns

The definition of a safety or mission critical, real-time, embedded system has to satisfy and preserve some crucial properties, such as:

- Operational and Functional behavior expectations as seen above (e.g. functions & services to be supported by the system, operational performances, operational use cases)
- But also non-functional constraints, whose a sample list might be
  - Technical performance,
  - availability and fault tolerance,
  - integrity,
  - security (confidentiality, resistance to attacks),
  - interface optimization,
  - maintainability, testability,
  - cost, weight, power consumption,
  - product policy,
  - but also ease of integration, ease of re-use, ease of extension…

Yet architecture, when designed only to solve one of these constraints, is often unsuitable to satisfy other expected properties (e.g. design rules favoring integrity / availability often lead to a degradation of performance and cost); modifications in order to take into account a given constraint may in turn bring bad adequacy to other constraints.

At the end, the selected architecture often appears far below expected operational usability, because essential criteria of satisfaction have been forgotten in front of definition and development constraints.

The technical problem to be resolved thus consists in facilitating a “multi-parameters” optimization of the architecture under possibly contradictory constraints, thus in bringing out an acceptable compromise between these constraints.

3.2 Adopting a Multi-Viewpoint approach

In order to properly address these constraints, ARCADIA adopts a viewpoint-based architectural description, such as described in standards [2].

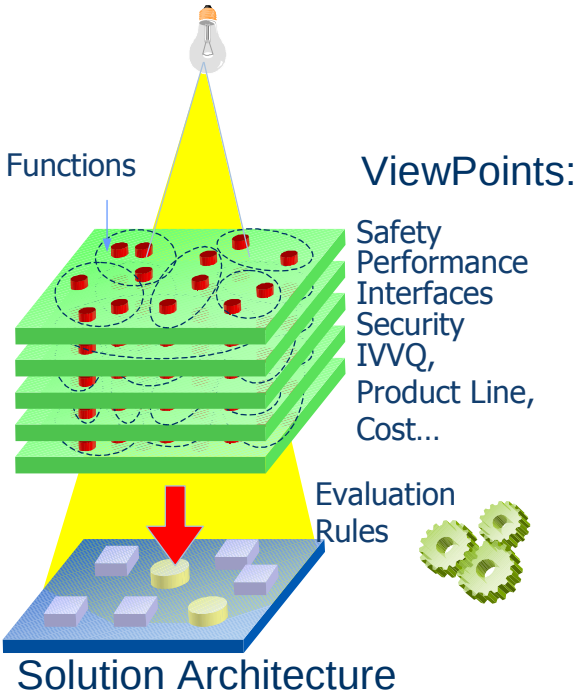
In this context, a view is “a representation of a whole system from the perspective of a

related set of concerns", while "a viewpoint defines how to construct and use a view".

Each major engineering concern likely to impact product definition and architecture design (see sample list above) is subject to viewpoint modeling: a dedicated viewpoint is created for each concern, so as to

- collect constraints, expectations, figures, related to the viewpoint, in the need analysis described above
- define means to address these constraints and design architecture to fulfil them
- submit each candidate architecture to these constraints, illustrating the way the architecture is involved and deals with these constraints
- define analysis rules to check that the solution fits expectations for this viewpoint.

Fig. 6. Multi-Viewpoint Architecture Analysis



Note that by this means, each stakeholder will address its own dedicated viewpoint, and confront it to others, in order to reach a collaborative architecture design and assessment.

3.3 Capturing non-functional need & requirements

Non functional requirements are analyzed and "translated" by decorating operational and functional need models with expected non functional properties, according to each viewpoint.

- As an example, at operational need level,
  - *safety requirements will be composed of 'feared events' associated to activities and exchanges between actors involved in the target mission*
  - *performance requirements will define expected reaction time to a given threat (detection between collision risk detection and avoidance maneuver)*
- At system need analysis level,
  - *system behavior that might lead to these feared events (e.g. functional chains, processing or display functions…) will be tagged according to the relevant criticality,*
  - *former reaction time will be decomposed and allocated to some system functional chains in terms of maximum acceptable latency.*

3.4 Checking Architecture candidates against non-functional constraints

For each viewpoint, some architectural patterns are applied to adapt architecture to these non functional requirements: *e.g. redundancy paths, multi-processing, among many others,*

Thanks to the continuousness and mapping of functional and non functional [need] models on architecture definition, the resulting architecture model can easily be analyzed in

order to check compliance with expected properties.

E.g.

- *Functional chains that are safety critical can be checked against redundancy issues (how many times are they allocated to different components in parallel?), common modes (are underlying components diverse or similar (thus introducing common failures)?); failure propagation and its functional consequences can be studied and simulated according to dysfunctional behavior model of components and functions (part of the safety viewpoint)…*
- *Time critical functional chains and processing implementation is tested, so as to estimate their execution time: thanks to functional complexity evaluation, and computing component resource estimation, an estimate of their latency can be computed, depending on the means they have been allocated to these execution resources; same estimation can be done on communication channels depending on functional data exchanges, etc.*

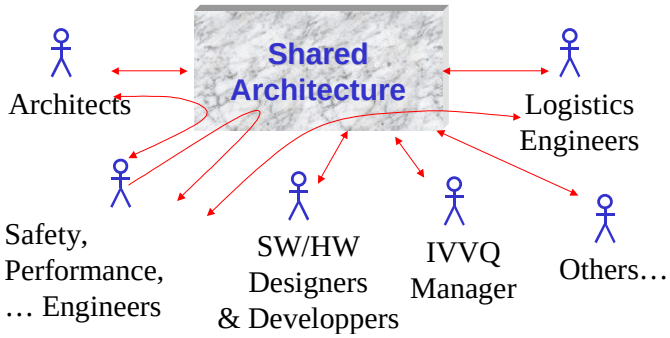
Candidate product architectures are therefore checked against all these viewpoints, simultaneously, and at each step of architecture building and assessment.

This is assisted by means of automated, **domain-dependent engineering rules checking** on models, and impact analysis of each architecture design choice.

It is to be noted that by this way

- Each engineering specialty has means to express not only its constraints, but also its architecture checking rules and solutions on the single, common model
- And that possible mismatches with its own golden rules will be automatically tracked and can be detected for each architecture design decision, *even if this design decision is done by other stakeholders*.

Fig. 7. Shared Architecture building & validation



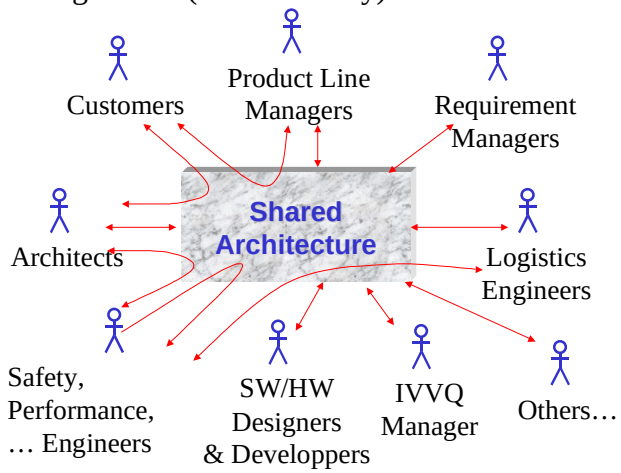
4 Model-centric collaboration

A third advantage of Arcadia is the ability of all stakeholders to **share a unique description of the product, its need and most engineering assets** described above; this is achieved through the collective use of a single reference model, that can be checked for completeness and coherency, and allows an efficient multi-user collaboration support and impact analysis:

- customer needs and models are captured and linked to product definition, for safe and consistent incorporation in the product
- architecture definition and first-level behavioral models can be shared as appropriate
- all stakeholders of system definition and assessment can share the same model and work together, while checking the consistency of their design decisions (through former viewpoints)
- transition and consistency between each engineering level (e.g. system/sub-system/software or hardware) is eased by automatic model transformation, bi-directional impact analysis

- sub-contracting is secured thanks to technical contracts based on architecture models
- integration and validation verification tests can be specified based on the need and product models above, and their results can be checked against these references
- finally, all the engineering data, assets and justification material can be efficiently capitalized and shared, including domain-specific know-how, reusable patterns and architectures…

Examples of roles that should cooperate in engineering can be (titles can vary)



- Chief architect
- Customer
- Operational expert
- Functional analyst
- Systems engineering manager
- Specialty engineering expert
- System engineer
- IVVQ[1] manager
- Configuration manager
- Software/hardware specialists
- Sub-contractors
- Product line manager
- Program manager

Some kinds of collaboration between engineering stakeholders, supported thanks to model sharing, can be quickly illustrated as follows: either both actors work collaboratively on the same model, or one of them produces entries in the model for the other one.

4.1 Customer – supplier collaboration

Customer need analysis model is intended to be built jointly between customer and supplier: at least they can share the operational view describing the end users goals, tasks and activities, along with operational scenarios that will be used for validation purpose.

Functional and non functional analysis can also be submitted to customer, in order to demonstrate how the system will meet operational requirements.

Then the supplier sketches a first allocation of functional analysis on an early architecture, in order to check feasibility. Here again, in case of some requirements being costly or complex to fulfil, the model can help in defining operational consequence of their modification or cancellation.

In some cases, if the customer also has a model-driven engineering, parts of the physical architecture of the expected system can be supplied as well, for larger scope validation (top level system early integration) at customer level.

4.2 Need analysts – architecture designers collaboration

Need analysts (operational experts, functional analysts) define the basic functional expectations on the system ; then the architects allocate these functions on architecture components, while preserving traceability and justification links with need model.

This formalization of need, solution and links relating them, is the basis for impact analysis: when a new requirement comes, when new functions are to be supported, these links are used to determine which parts of the architecture are to be modified, and under which conditions, thus enabling cost & risk assessment among others.

[1] IVVQ : Integration, Verification, Validation, Qualification

4.3 Specialty engineering experts – chief architect collaboration

Specialty engineering experts deal with concerns regarding a specific field, such as safety, performance, interface management, security, logistics & support, maintainability, product line issues etc.

Most of them need to check architecture design against their own constraints; furthermore, in many cases, the architectural design would take great benefit from anticipating these constraints in an a priori design, instead of an a posteriori compliance check. And of course, if each one developed his/her own model, this would lead to discrepancies and rework.

All these expected features are achieved and secured through ARCADIA single model sharing. Three means to support this collaboration are currently put in practice, mainly through the viewpoint-driven approach described above:

- Specialty-specific architectural patterns can be put at disposal of the architect in the model, in order to fulfil non functional constraints (e.g. safety barriers, multi-processing patterns, middleware features…).
- Specialty-dedicated viewpoint rules can be used to analyze and check the architecture from one specialty point of view, at each elementary design decision, and even if the specialist is not present (thanks to viewpoint rule checking).
- The common model can also be used to feed specialty-dedicated tools: such as. quantitative simulation, safety analysis (fault tree, minimal cuts…) or 3D spatial layout. Possible extensions and complementary descriptions provided by these dedicated tools are added to the common model, so as to preserve coherency and capability of round trip.

4.4 System engineers - chief architect collaboration

Model building and check can be split between multiple users, depending on product breakdown and enterprise organization. But each develops or analyses only a part of a common model, so as to ensure consistency and uniqueness of product building.

Note that various kinds of work share can be used: e.g. split by phase (operational analysis, functional analysis, architecture), by functional contents, by component, or by viewpoint.

4.5 Supplier - Sub-contractor collaboration

Requirements for suppliers of system components (sub-systems, equipment, software, hardware) are deduced from the physical architecture model, based on allocation to the components they are responsible for.

The architecture model allows to largely enrich traditional requirements and interfaces and turn them into a real “technical integration contract”, with functional contents, expected operational behavior (through operational analysis and scenarios allocation), non functional constraints allocation (resource consumption, quality of service…), all these information being deduced from the ARCADIA model.

If each component respects this integration contract, then the integration, verification and validation (IVV) phase is likely to be easy and straightforward.

4.6 System – software architects collaboration

In the case of software components, the integration contract can go further and turn into a preliminary software architecture model, defining expectations on software.

This can take the form of UML models, delivering data model, components models, sequence diagrams etc, that the software team will refine, detail and organize into a software architecture fully traceable and checkable against the system-level physical architecture.

4.7 Integration/validation manager–Architect collaboration

Contents and policy of the integration, verification and validation (IVV) phase can partly be defined based on architecture model: e.g. the contents of each delivery can be determined from desired operational capabilities, activities; test cases can be derived from operational scenarios; dependencies in deliveries can be checked accordingly, and in case of missed deadlines, consequences on available functions can be determined easily.

5 ARCADIA supporting Tools

Arcadia supporting tools are crucial for best benefit from the method, both because they help in managing complexity and size of shared information, and support collaboration between stakeholders, along with early validation and justification issues.

- they must allow and ease capitalizing models, concepts, engineering rules and architectural assets,
- while adapting to each domain for specific extensions and enrichments.
- They also have to manage multi-user issues (configuration management, shared model access, intelligent diff/merge…) so as to take real benefit from the common reference model.

5.1 Basic engineering support features

In order to support model-driven engineering activities, the toolset supporting ARCADIA, named ORCHESTRA, running over Eclipse (see [4]) delivers the following common, widely spread functionalities:

- Modeling editor with enhanced graphics / diagrams capability, syntactic checking…
- Semantic model browser
- Model transformation & transition support tools (for system to sub-system, to software and to hardware engineering transition)
- Model import/export means (including towards excel, access and dedicated specialty engineering tools, product lifecycle management and more)
- Requirement management tools
- Version and configuration management tools, coupled with model repository, data management
- Documentation generation tools (from model)
- Link manager for model elements and engineering assets, traceability and impact analysis means
- Test & simulation support (for models, including test scenarios definition, run, analysis)
- …

Note that maximum proximity with state of the art concepts have been preserved each time it was possible, therefore allowing interoperability with standards such as Architecture Frameworks (see [5]), UML and SysML (see [6]) and AADL (see [7]).

5.2 Method dependent extensions & adaptation to dedicated domains

The heart of ARCADIA model-driven approach in ORCHESTRA toolset is an enhanced architecture modeler/checker called MELODY ADVANCE.

Beyond basic modeling capabilities, many features are necessary to achieve ARCADIA benefits:

- modeling and complexity management aids
  e.g. complexity hiding, automatic synthesis, automatic diagram creation…
- ability to enrich and extend ARCADIA basic concepts (so called ‘meta model’) for specific domains and specialty engineering
  e.g. safety concerns such as feared event or development assurance level, IVV versions, …
- ability to customize existing

diagrams and create new kinds of diagrams (with a Domain Specific language DSL) for dedicated analysis
e.g. automatic impact analysis and traceability support diagrams, dependency diagrams…

- ability to define model analysis and check rules, as needed for each viewpoint
  e.g. safety or performance compliance checks
- multi-viewpoint compromise analysis tools
  e.g. rejection criteria if safety issues are not preserved
- capitalization and reuse support
  e.g. reuse libraries and checking viewpoints, architectural patterns management …

All these features developed for a given specialty engineering, are packaged and managed as a whole, in one or several viewpoint support packages (e.g. safety viewpoint, performance viewpoint, IVVQ, cost, reuse… viewpoints).

5.3 Multi user collaboration support

Multi-user sharing is currently based on a simple check-in / check out mechanism, each user defining the model parts to be modified, and then locking them until modification completion.

This is applicable in case of partitioned, tree oriented breakdown, but it appears that architectural building and validation is not necessarily of this kind: as an example, analysis from one specialty engineering point of view is hardly decomposable this way.

This is why an other way of collaboration is being developed: it is based on live sharing of one single model, and instantaneous, atomic locking.

This greatly helps in reducing complex model comparison ("diff and merge") issues, that are otherwise still necessary in order to compare non-synchronized evolutions of different stakeholders. These model comparison functions will yet be enriched in order to better identify evolution outline and intents, especially for multi-branch management purpose.

6 Conclusion and future work

Arcadia is currently in use in Thales pilot programs, and supported by Thales Orchestra Tool suite.

The benefits already shown by these experiments already appear to be:

- Less rework in design & production thanks to Early validation of key architectural aspects
- Efficient support to decision making regarding complex but necessary architectural trade-off
- Ability to capitalize both product definition, know-how, and decision making

- Support to negotiation and compromise between stakeholders
- Support to interoperation with Customers & Suppliers

- Ability to adjust modelling effort:
  - scaled/focused on major engineering issues for return on invest
  - Without exhaustive modeling.

Future work is split between adaptation to each domain and viewpoint, as mentioned here above, and tools & processes enhancement; among others:

- in the near future, multi-user and configuration/evolution enhanced support; link with IVVQ phases (in progress), transitions towards software and hardware, application to hardware
- in the mid-term, architecture design aids for solution emergence, modeling

automation aids, integration with simulation…

7 References

[1] Method & Tools for constrained System Architecting, Jean-Luc Voirin, at INCOSE'08 Symposium

[2] ISO/IEC 42010:2007 (also known as IEEE Std 1471–2000) Systems and software engineering - Recommended practice for architectural description of software-intensive systems

[3] Méthode APTE, http://www.methode-apte.com/methode_apte.htm

[4] Eclipse and EMF modelling technologies, http://www.eclipse.org/modeling/emf/

[5] NATO C3 System Architecture Framework (NAF), AC/322-D (2004)0041, NATO C3 Board, 2004

[6] Systems Modeling Language (SysML) Specification, OMG document: ad/2006-03-01, 2006

[7] Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment, SAE ARP 4761, December 1996

8 8 Contact Author Email Address

jean-luc.voirin@fr.thalesgroup.com

9 Copyright Statement