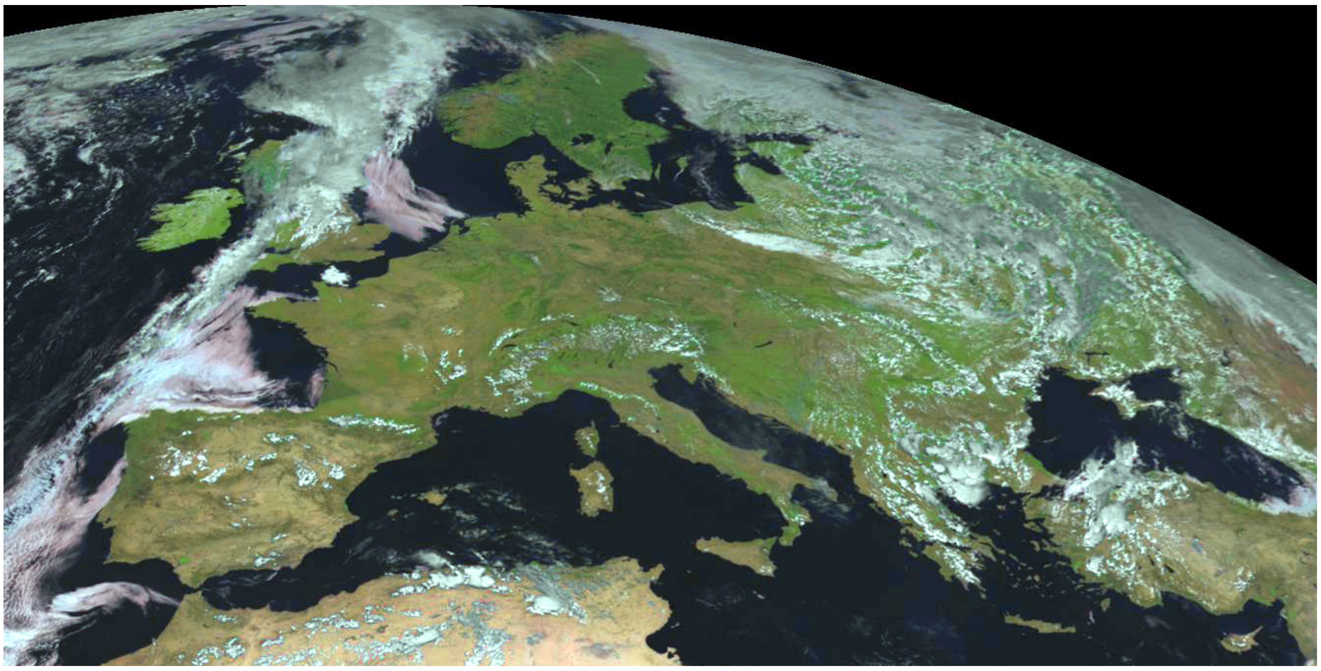
Abstract Book
&
Final Programme



ESA Workshop on Avionics Data, Control and Software Systems (ADCSS)

29 - 31 October 2008

ESA/ESTEC
Noordwijk, The Netherlands

Organised by the European Space Agency

Table of Contents

Programme Committee

Philippe Armbruster	Data Systems Division
Alain Benoit	Control Systems Division
Kjeld Hjortnaes	Software Systems Division
Roger Jansson	Control Systems Division
Juan Miró	Software Systems Division
Patrick Plancke	Data Systems Division
Jean-Loup Terraillon	Software Systems Division

Workshop organizer

Andreas Jung	Software Systems Division

Round Table organizer

29th October: Space Avionics Open Interface Architecture (SAVOIR)

Andreas Jung	Software Systems Division

30th October: Model Based Software Engineering

Yuri Yushtein	Software Systems Division

31st October: ISVV Process Improvement

Sabine Krüger	Software Systems Division

ADCSS Programme

Space Avionics Open Interface Architecture (SAVOIR)

Wednesday, 29 October 2008

08:30 Registration

09:00 Introduction

ADCSS Programme

Model Based Software Engineering

Round Table
16:30 - 18:00

- Experience and lessons learnt from the use of modelling techniques in onboard software development
- Maturity of current methodology and tools in support of the model based software engineering (SysML, [HRT-]UML, OCL, assertions, AADL, SDL, SCADE, Matlab, etc), and how to apply them to the onboard software development process
- What is the new (model) life cycle and how does it impact current practices?
- What is the current coverage of this process by supporting methodology and tools?
- What is the added value of deploying fully or partly the Model Based Software Engineering process wrt the current practices for onboard software development?
- What is blocking us from a wider use of model-based approaches? (maturity, coverage, cost,…)
- What R&D actions are required to facilitate the adoption of such a process?
- How to ensure model consistency along the life cycle of avionics and software? Is there a life cycle for models?

ADCSS Programme

ISVV Process Improvement

Results of the COeDrT (Component Oriented Development Techniques) and DOMENG (Domain Engineering) activities

Andreas Jung (ESA) on behalf of the three studies

In the ESA Harmonisation exercise in 2006/07, the need arose for the definition of space avionics/embedded reference architecture in general (hardware, software and communications), and software reference architecture in particular.
In that landscape, three industrial contracts were started in early 2007 by ESA (COrDeT-Astrium (Toulouse), COrDeT-Thales (Cannes) and DOMENG) with the task to use the domain engineering process to sketch a reference software architecture for the domain of interest. The project teams include EADS/Astrium-Toulouse, GMV, Intecs, the University of Padua, P&P Software, SciSys, SoftWcare, and Thales-Alenia Space/Cannes. The scope of the space domain to be analysed is Space Avionics/embedded systems for earth observation, science, telecom, navigation related missions.

In conjunction with the final reviews of the two Cordet studies end of September, an internal workshop was performed. The outcomes of this workshop will be presented in this presentation and will provide the status, results and way forward of the COrDeT and Domeng studies.

ASSERT Exploitation Plan

Eric Conquet (ESA)

The ASSERT project initiated in 2003 and ended in 2007 has merged together the motivation and expertise of around 30 partners in Europe. Partially funded by the European Commission under the FP6 umbrella, this Integrated Project targeted the definition of a new development process for real-time and software dominant systems with modeling and proof as the main technological pillars.

After three years of hard work, ASSERT has successfully delivered a complete and consistent tooled process whose efficiency has been demonstrated on some close to real case studies. Although the tools are still in the prototype phase, and despite the youth of the process and lack of experience of engineers used as guinea pigs, early figures indicate a boost of around 25% of the productivity.

Following this first experience, we need now to prepare the future and take all lessons from ASSERT. The exploitation plan is addressing all action types to be initiated, all objectives that the ASSERT open community can draw, and all possible funding opportunities. This results in a set of proposals and a possible roadmap addressing the entire scope from introduction of some technologies into operational projects to the preparation of larger initiative to extend the capabilities of the ASSERT process and tools with the ultimate goal of bringing innovation to the desktop computer of our engineers.

Time and Space partitioning

Kjeld Hjortnaes (ESA)

Within the aeronautic industry the Integrated Modular Avionics (IMA) has successfully been deployed to facilitate the significant increase in software functionality and at the same time support a higher level of avionics system integration to save mass, volume and power. It has been deployed on the new generation of high end aircrafts as the A380 and the Boeing-777 / Boeing -787 and also in many smaller aircrafts.
The IMA concept provides an integrated architecture that retains the fault containment properties of the traditional federated architecture. This is achieved by introducing Time & Space Partitioning of the software within the shared computing resources. This segregation principle among others allows independent functional chains to be developed and validated in isolation, which has significant advantages in the system integration phase of a project.
The topic of transferring the IMA concept to space application was extensively discussed at the ADCSS-2007 workshop, and as a consequence of this a joint working group was established between ESA / CNES / Astrium- Satellites / Thales Alenia Space to study the concept, identify the potential benefits to the space industry and identify the User Requirements that consequently would be imposed on the software or satellite design.

The presentation will report on the work done within this working group and in particular will present a number of use-cases from the satellite domain where the IMA architecture could be beneficial. The presentation will also report on the technology activities required before a full scale deployment into the satellite design can be supported.

Additionally, to get an overview of on-going TSP technology activities a set of short overview presentations will be given of the on-going studies within the IMA/TSP topic.

1. ESA: AIR-II: Arinc653 complaint RTOS (Skysoft/ University of Lisbon /ThalesAlenisSpace)
2. NLR: Study of I/O in Time and Space Partition
3. CNES: ARINC study (Astrium)
4. CNES: XTRATUM – A microkernel for TSP system (UPV – Valencia)

AIR-II: Arinc653 complaint RTOS (Skysoft/ University of Lisbon /ThalesAlenisSpace)
James Winsor (ESA)

AIR-II is an activity with Skysoft started in Dec 2007, which continues the development of the proof-of-concept ARINC 653 compliant Operating system (based upon RTEMS 4.8). The AIR activity defined the base design of an ARINC 653 RTOS kernel using a multi-executive RTOS core and two-level hierarchical scheduler architecture. The activity implements: the AIR Partition Management Kernel (PMK) supporting partitioning in both spatial and temporal domains and the ARINC 653 APEX interface (the application executive interface at each partition), the LEON2 support, the MMU support to RTEMS, tackle the lack of user mode support in RTEMS, support the deployment of any RTOS (including commercial RTOS kernels) within the partition, and involve a systems integrator (TAS-F) to bring avionics domain expertise to the activity. The intention is to demonstrate the feasibility of using a ARINC 653 compliant RTOS on a space on board data system. The demonstrator application will be developed using the Skysoft AMOBA simulator in order to ensure conformance to ARINC 653. The outputs from the AIR II activity will a ARINC-653 for Space RTOS, user documentation, validation evidence, and an evaluation of the relevant changes to be done to the ARINC 653 standard to make it more compatible with the space domain.

Study of I/O in Time and Space Partition (NLR)
Guillaume Francois (NLR)

I/O handling is considered one of the major concerns in introducing the Time and Space Partition into the Space application domain. ESA has initiated a study to in detail analyse the I/O handling strategies used in the aeronautic domain and assess the impact of such strategies if utilised within Space Systems. It is the assumption that TSP can be introduced with minimal hardware impact. Functionality required are basically present in ERC32/LEON processors including block protection functionality. The block protection can be provided by dedicated electronics or the Memory Management Unit – MMU.

CNES: ARINC study (Astrium)
Paul Arberet (CNES)

The study contracted to ASTRIUM, consists first in catching the lessons learnt from ARINC, and to a certain extent from IMA, in the scope of the aeronautic industry. The main outcome from this phase was the identification of the pros and cons of the ARINC 653 standard, with in particular an identification of the missing key aspects for the development of a TSP system and consequently to a partitioned OS.
The second part of the study, still on-going, aims at making the transposition in the space domain context. In particular the ARINC 653 and the additional missing requirements have to selected for space applications. Some implementation guidelines and architectural key choices have to be assessed and consolidated in order to propose, at the end, strong recommendations for the future.

CNES: Xtratum – A microkernel for TSP system (UPV – Valencia)
Paul Arberet (CNES)

In the scope of scientific payload development, CNES decided to invest in TSP approach, which should enable to decouple generic protocol, monitoring software from mission specific functions. Cost and technical rationales conducted to analyse the Xtratum hypervisor opensource solution, developed by UPV (Valencia). The on-going activities with UPV are consisting in porting Xtratum on LEON2 target and assess the validy of the concept in the stringent space context, in particular wrt memory and CPU footprint. First results are very promising. The second phase of the activities will aim at fully prototype and integrate on a representative use case the complete solution including : "high level OS" such as RTEMS, middleware architecture components (including common services), I/O partition, TM/TC -PUS- application and services.

Status of standardisation for Data Systems
– On-board communication services and protocols
– Microprocessors, IP Cores
– Communication buses and links
Ph. Armbruster (TEC-ED/ESA), C. Taylor (TEC-EDS/ESA) and other members of the Data Systems division

Data systems and in particular on-board computers have been framed already for a long time by standards. This is witnessed not only by TM/TC protocols; standardised in an early stage to ensure a good coherency between space and ground segments and at inter-Agencies cross-operations level but also by on-board communication buses such as the Mil1553B and to a lesser extend the OBDH bus. A similar approach drove the selection of on-board processors. Being based on a well defined instruction set (e.g. 1750) or an architecture (e.g. SPARC), they refer to de-facto standards in order to preserve SW investments with a suitable compatibility for SW applications through high-level languages and this across processor families. More efforts were deployed on this foundation by Agencies and Industry to apply an even more systematic approach. This led to considering the standardisation of services on top of the underlying on-board communication infrastructure. From this point of view the PUS standard has been a precursor that has been complemented by other presently defined CCSDS services (i.e. SOIS).

Once functions are being defined and interfaces specified, building blocks can be designed as components (such as the ERC32 and LEON-2 processors, Mil-155B BC/RTs, SpaceWire link controllers and components like the SCTMTC), or as IP Cores that foster re-use in SoCs.

The presentation will follow in its first part by a top-down approach by providing the general objectives of data systems standardisation and the methodology proposed to define building blocks when linked to reference architectures. This will be followed by a status review of developments related to On-board communication services and protocols, microprocessors, IP Cores and finally communication buses and links.

Study of the automotive architecture standard AUTOSAR for embedded systems

William O'Connor (Rovsing – Ireland)

Rovsing Ireland Ltd has lead this study in collaboration with our academic and industrial associates. The automotive community recognised that a level of complexity in automotive systems had been reached which was difficult to handle using traditional development processes. Resources spent on adapting existing solutions to different environments, an increasing number of networked components, increasing demands in vehicle safety, reliability, driver assistance and comfort demanded a technological breakthrough. AUTOSAR (AUTomotive Open System ARchitecture) was developed as a key technology to address these challenges.

AUTOSAR is an open and standardised automotive software architecture. AUTOSAR was founded in 2003 by OEMs and Tier 1 suppliers and now includes a large number of electronics, semiconductor, tool supplier, hardware and software companies.

The resulting modular software architecture shares many of the features and addresses many of the concerns addressed by the highly reliable real-time software architectures that are required in the aerospace and space industries. As a result of this, even if AUTOSAR is not directly appropriate, it may be possible to employ AUTOSAR concepts in terms of technical and organisational solutions. The economics of the space industry mitigate against such a large effort being applied for a similar solution designed specifically for space applications.

Initial concerns about the suitability of AUTOSAR have been raised because it does not address exactly the same concerns as those that are faced in spacecraft onboard software. Specifically, there is some concern as to what degree the existing AUTOSAR standard can support the development of safety and time critical software. Furthermore, the lack of a centralised fault management framework and partitioning concept in the current AUTOSAR standard must be evaluated against the future requirements of spacecraft on-board software.

The aim of this study is therefore to resolve many of the unanswered questions surrounding AUTOSAR and to determine if it can be applied in whole, or in part, to spacecraft software development.

Model Based Development and Testing

Mazzini, S.; Mambrini, R.
Intecs SpA

Intecs has experience in the production of embedded and critical SW systems, as well as on SW engineering, not only in the Space, but also in the Automotive, Railways, Telecommunications and Defense domains. Moreover Intecs has a large experience in V&V in all these domains and witnessed the evolution of SW development and the change of needs in testing, due to the introduction of models in lifecycle.

Nowadays the practice of SW produced by handwritten code is decreasing, and it is usually limited to the HW dependent part as the SW is more and more automatically generated starting from models.

In the classical approach, unit, integration and system test techniques are usually combined, and test procedures are usually handwritten, only coverage tests can be partially automated using specific commercial tools.

SW programming scenario is actually changing because of the use of models, so the programmer role is changing as well: the focus is on the development of models, the code is automatically generated from these models. The abstraction level is higher than in the past.

In this new scenario also the testing methodology changes: the need of testing the model is arisen. If the model is verified, and validated against the SW requirements, and the code generator is validated or proven in use, then we can state that model V&V, achieving the model coverage, can largely replace (manual) unit and integration testing.

V&V of models requires combining a dynamic approach, by animation or simulation, together with static analysis techniques.

V&V may be applied to the different view/diagrams, depending also on the applied methodology:
-Interaction scenarios are associated to use cases to describe use or component interaction , in relation to SW requirements, to help to verify the system model behaves according to the specification (model validation)
- State machine diagrams show the SW components functional behavior and state evolution related to events. Formal verification as well dynamic approaches can help to verify the system behaves correctly (model verification)
Starting from accurate models, model-based testing may control automatic test generation. Scenario-based and state machine-based testing approaches can be combined by defining an integrated and practical method for the strategic generation and planning of UML-based test suites.

As example of this new 'modus operandi' we can look at the experience Intecs is acquiring as AUTOSAR premium member, participating to the Conformance Tests WP. Conformance tests goal is to demonstrate the adherence of Base SW modules to the AUTOSAR standard. These modules are developed as UML models and these models are used as inputs for the definition of conformance tests written in TTCN-3 language.

Model based design and testing are going to change the development process, models become more and more important actors, the classical V development model should be extended to accommodate both the model and and SW life cycle.

Application of Model-Driven Engineering to the Development of On-board Software: Benefits and Challenges

Panunzio, M.; Vardanega, T.
University of Padua

The validation-intensive nature of high-integrity real-time systems makes the adoption of Model-Driven Engineering (MDE) less obvious than in other industrial domains. In this presentation we discuss some of the obstacles to remove.

Dispelling MDE fads
For MDE to be intelligently used for high-integrity real-time systems, some essential characteristics must be actively sought: (i) *separation of concerns*, to help designers acquire intellectual control on the aspects of their specific pertinence without the need or the pretense to master all system and software aspects; and (ii) *correctness-by-construction*, to warrant proactively (as opposed to a posteriori) the consistency of all the models that compose the system and to ensure the preservation of properties throughout the development until run time.

Models are the means to promote and attain focus of attention (hence simplification) and abstraction. The use of a graphical notation per se is neither sufficient nor desirable, unless it increases the expressive power availed to the designer. Creating models may otherwise become as complex and error prone as writing code in (unfamiliar) languages. The abstraction level availed to the designer may be raised by favoring the use of libraries of pre-built components (tantamount to the reuse of models) as well as of *MDE-geared design patterns*, which address and provably solve recurrent domain-specific problems in a fashion coherent with principles (i) and (ii) above.

Highlighting the added value of MDE
In an on-board system, parts of the code are inherently *archetypal*, thus rigidly defined and with little or no implementation freedom (e.g., concurrency-related code). Other parts leave more space to expert skills (e.g., control-related algorithmic code). An MDE incarnation suited for the space domain should have designers solely focus on the parts that benefit from their expert contribution; all other aspects should be left to proven automation. That decision should earn increased productivity and less defects.

Coping with heterogeneous models
The models used and produced by the various actors in the development team are likely to be heterogeneous in scope, content and formalism. Models may address *different* aspects (often called "views") of the system and also be the product of view-specific formalisms and technologies. A possible yet undesirable consequence of this situation is that models may be incompatible in either technology or (worse) semantics. The most feasible way to defeat this risk is to acknowledge heterogeneity and require that models be provably *composable*: this is easier to ensure if all metamodels that underpin the user models address *non-overlapping* aspects (and thus separate concerns).

Potential Benefits of MDE for Space Engineering

Poupart, E.
CNES

In this paper, we present the participation of the CNES in a research project called DOMINO financed by the French National Research Agency (ANR) RNTL, through a case study. This project started in March 2007 and will end in March 2009.It regroups academics (ENSIETA, IRISA and IRIT), industries and agencies (AIRBUS, CEA, CNES and SODIFRANCE). The project has two main goals: to develop reliable Model Driven Engineering (MDE) components and to build bridges with Domain Specific Languages (DSL).

CNES participates in this project through a case study on the reliable design of operational procedures and associated applications. There are two main objectives for this case study:

The first is to improve "offline" interoperability with the possibility to build import/export tools for any scripting procedure language using MDE transformations.

The second is to inject the use of models into the ECSS Space Segment, Ground Segment and Operations Engineering Processes to improve efficiency, reusability, reduction of costs, and reliability.

We will describe how it is possible to take into account some specificities like ECSS-E-10 "supplier/customer" model, how textual requirements including informative and prescriptive statements should be considered in a model driven engineering approach, how specifications (what) and design (how) are interrelated in a multi-level system especially for the derivation and allocation of requirements, etc.

We will illustrate these elements on a subset of the processes for the "Star Tracker" subsystem.

Model Driven Engineering at EADS Astrium Space Transportation

Lesens, D.; Blanchet, G.; Dalemagne, D.;
Gast, Ph.; Zindy, G.
EADS Astrium Space Transportation

This presentation will describe the current status of Model Driven Engineering (MDE) at Astrium ST, from capture of system requirements allocated to SW to automatic code generation (ACG). This presentation will focus on Mission and Vehicle Management (MVM) SW (GNC is covered by another presentation). It will present the MDE approach already used on operational project and the current R&T studies.

Operational project
SART has been used on the ATV nominal SW (FAS) to describe the system functional architecture and automata. These automata have been simulated in order to perform an early SW specification validation (AIFV). The SW specification document has been automatically generated. The approach main advantage is an increase of the SW specification quality. The drawbacks are the non support of hierarchical automata and of sequence diagrams and a weak semantics for real time constraints and data.

Scade has been used for the specification of the ATV safety SW (MSU) and of the M51 and Vega SW. Simulation and formal proof have been used to improve the quality of the ATV MSU specification. Part of the documentation has been automatically generated. The advantages of the approach are identical to the ones of SART with the additional advantage of a stronger semantics (and then a better quality) than SART. The main drawback is that Scade is not adapted to model abstract requirements and thus the use of Scade shall be completed by other models (generally in UML). Moreover, the used versions of Scade (V3 to V5) were not adapted to automata ACG.

R&T
Two main tracks have been followed to improve the MDE approach at Astrium ST: the use of UML/SysML and of the last version of Scade (the use of Simulink for GNC is not covered by this presentation).

Rhapsody UML/SysML has been used to capture system requirements allocated to SW on an Ariane 5 case study. A single model (using the MARTE profile) has been developed in co-engineering between vehicle system design and SW teams. It has been simulated and allows generating the system design documentation (DF-05) and the SW specification (ST), in a manner very clear and understandable even by non UML experts. The advantages are an improvement of the consistency of the documentations (generated from a single checked model) and of the quality (validated by simulation). The natural follow-up is the ACG from class diagrams and automata. The studies performed at Astrium ST have shown that ACG for class diagrams (in C or Ada) is mature, but that ACG for automata is not.

Scade V6 has been used to model the most complicated ATV automaton (the propulsion). This model has been validated by simulation and is then usable to generate a clear documentation (SW specification) understandable by non Scade expert and for ACG.

The presentation will conclude by links between MDE and the internal development process (MELANIE) and the perspectives of use of the MDE approach in future operational projects.

Model-Checking and Validating UML Models: Current Capabilities and Limitations

Faria, J.; Silva, N.; Brandão, P.; Esper, A.; Matoso, M.; Mahomad, S.
Critical Software

Model-checking and validating UML models are subjects of current thorough investigation, both in Academia and Industry. Notwithstanding, most of the work found in the literature describes (theoretical) work using small examples for the validation of concepts. We present the results of a project where UML, SysML, OCL, and Model-Checking are combined for the modeling and validation of a large set of software system requirements.

Model-checking is a technique gaining more and more acceptance in Industry. It is a very powerful mechanism for the validation of the dynamic behaviour of systems, especially concurrency issues. We combine model-checking and UML by using a tool, Hugo/RT, that automatically translates UML models into Promela, the input language of the SPIN model-checker. We thoroughly access the maturity of such methodology (including the applied and other related tools), evaluating the capabilities and limitations of the translation process, purposing possible solutions/workarounds and pinpointing research directions in such field.

In the project, OCL (Object Constraint Language) is used for the static validation of the UML/SysML models. The approach is to define a set of rules that can be written as OCL expressions applied to the meta-entities of UML/SysM. Like that, the rules stay perfectly general and may be applied in any project. We access both the maturity of UML tools to support such approach and the suitability of the method for the validation of the models.

Finally, other issues addressed are the possibility of using OCL to validate dynamic properties of models and the possibility to automatically extract tests from the developed UML/SysML models.

Operational use of MDE Process and Component Models at Thales Alenia Space on the GlobalStar2 Project

Garcia, G.; Le Coroller, M.; Moreno, C.
Thales Alenia Space France

Following studies presented in a paper at DASIA conference 2003, the concept of modular and plug'n'play architecture has been refined by Thales Alenia Space France in the scope of its IDL Compilers and Software Bus products. Thales Alenia Space France IDL Compilers and Software Bus deal with topics related to the definition of reference on-board software architecture, and the standardisation of software component - aka software building block - interface definition.

This paper describes the studies performed in the scope of internal research activities, and provides an overview on the industrialisation and achievements of the IDL Compilers and Software Bus on the Globalstar 2 constellation industrial project. This paper is primarily organized in four sections.

It first deals with the origins of the IDL compilers and Software Bus. This section will quickly address the issues the Software Bus shall answer to, as for example the cost and planning reduction for the on-board software development inversely mirroring increase quality and complexity expectations. This section will also described how the standardisation of interface modelling in a component oriented architecture allows the definition of a reference architecture, as well as cost and error savings in the extra-functional code production, and in the integration of possibly subcontracted third-party components.
In a subsequent section, the paper addresses the proposed concrete basis for Thales Alenia Space France solution. It in particular covers the tailorization to the Space domain of an existing open standardized Component Model, the definition of transformation rules from the Component Model to Real-Time Embedded code for communication handling compliant with ECSS PUS, the production of code / Interface Definition Sheets / and documentation generators, and the production of the base execution infrastructure / middleware.
The use of the IDL compilers and Software Bus in a industrial context will be then assessed in the scope of the feedback from the Thales Alenia Space Globalstar 2 project, upon which these solutions are deployed.
Last but not least, tracks for future enhancement of this component-based Model Driven Engineering process will be addressed.

The AADL for Real-Time modelling

Dissaux, P.[1]; Hugues, J.[2]; Singhoff, F.[3]
[1]Ellidiss Technologies; [2]Telecom Paristech; [3]University of Brest

Version 2 of the Architecture Analysis and Design Language (AADL v2) will be released soon by the SAE (http://www.aadl.info).

The aim of the first part of this paper is to provide an overview of the language and its usages since its first issue in November 2004 (SAE AS5506) and annexes in July 2006 (SAE AS5506/1), quickly present current tool solutions and reveal the main new features of AADL v2.

AADLv2 is a standardisation effort, with the cooperation of avionics and space contractors. In particular, the ASSERT project contributed to its definition. AADLv1 proved to be a good vehicle to model, validate and generate code for mission-critical systems. AADLv2 corrects some usability aspects of the language while preserving most of its benefits.

In the second part, we will describe the way this international standard addresses real-time modelling issues and complex industrial life-cycle support.

As opposed to general modelling languages like UML, the AADL is a Domain Specific Modelling Language (DSML) that targets software intensive critical real-time systems. The AADL is not simply an extension of a general purpose language to support real-time paradigms. On the contrary, the core of the language actually focuses on real-time engineering requirements:
- it defines concrete real-time architectures as well as library of re-usable components
- it takes into account the run-time semantics through pre-defined categories of components
- it provides an exhaustive description of the instance model to enable advanced verifications

These three topics will be addressed in the paper and illustrated by examples derived from on-going AADL-related projects such as SPICES, FLEX-EWARE and LABASSERT:
- use of AADL packages and operational systems to implement modular real-time modelling life-cycles where separation of concerns can be expressed with the same language but in separate views (pure functional description, real-time software architecture, hardware infra-structure, deployment information, error model, ...), and later merged together into a concrete, consistent and exhaustive definition of the operational system.
- use of the AADL run-time semantics that allows the definition of pre-defined real-time modelling patterns such as synchronous data-flows, mutex-protected data, blackboards or queued buffers, which can highly simplify the task of real-time engineers while minimizing the risk of inconsistencies between the applicative model and the actual characteristics of the execution platform.
- use of the AADL textual description as a pivot language for analysis and production tools: due to its status of official international standard and its intrinsic capability to carry the exhaustive specification of a concrete real-time system, the AADL is becomming a reference as a model descriptor input for real-time analysis and production tools.

System and Software Co-Engineering: Performance and Verification

Bozzano, M.^{1}; Burte, G.^{2}; Cimatti, A.^{1}; le Coroller, M.^{2}; Katoen, J.-P.^{3}; Nguyen, V.Y.^{3}; Noll, T.^{3}; Olive, X.^{2}
^{1}Fondazione Bruno Kessler; ^{2}Thales Alenia Space; ^{3}RWTH Aachen University

We report on a model-based approach to system-software co-engineering which is tailored to the specific characteristics of critical on-board systems for the space domain. The approach is supported by a System-Level Integrated Modeling (SLIM) Language in which engineers are provided with convenient ways to specify a.o. nominal hardware, as well as software operations, (probabilistic) faults and their propagation, error recovery and degraded modes of operation. This language is strongly based on AADL and its error annex which allows for the modeling of error behavior. A kernel of the SLIM language is equipped with a formal semantics that provides the interpretation of SLIM specifications in a precise and unambiguous manner. Systems are considered as a hierarchy of (hardware and software) components where components are defined by their type (interface) and implementation. Components communicate via ports allowing for message and continuous communication. The internal structure of a component implementation is specified by its decomposition into subcomponents, together with their HW/SW bindings and their interaction via connections over ports. Component behavior is described by a textual description of mode-transition diagrams. System reconfiguration is supported by mode-dependent presence of subcomponents and their connections. Error behaviour is described by probabilistic finite state machines, where error delays may be governed by continuous random variables.

Correctness properties, safety guarantees, and performance and dependability requirements are specified using requirement specification patterns which act as parameterized "templates" to the engineers and thus offer a comprehensible and easy-to-use framework for requirement specification.

The properties are checked on the SLIM specification using formal analysis techniques such as model checking and probabilistic variants thereof. The precise character of these techniques and the SLIM semantics yield a trustworthy modeling and analysis framework for system and software engineers. The formal analysis is based on state-of-the-art model checking techniques such as bounded SAT-based and symbolic model checking, and extensions of model checking with numerical and simulative means to reason about quantitative requirements such as performance and dependability. The analysis facilities support, among others: automated derivation of dynamic (i.e., randomly timed) fault trees, FMEA tables, assessment of FDIR, and automated derivation of observability requirements.

The realization of an integrated platform tool set is currently under development. Evaluation of this novel approach to system-software co-engineering will take place by means of selected case studies representing critical on-board computer-based systems.

Requirements-based and Model-based Testing

Eschbach, R.
Fraunhofer IESE

Model-based software engineering requires models for verification and validation. Hence systematic techniques for the construction of models are very important. One may distinguish between two different types of models: environment models and system model. These models play a crucial role within model-based testing. Environment models can be used to stimulate the system under test. Test cases can generated automatically from the model description using different selection criteria. Of special interest are deterministic/stochastic environment models like a combination of Markov Chains and Mealy Machines in order to detect both anticipated and non-anticipated failures. The system model describes the expected system behaviour and can be used to build the test oracle. With these kinds of models the complete test chain can be automated: test cases can be generated automatically from the environment model, test cases can be automatically executed and automatically verified against pass/fail criteria using the system model. A complete test result analysis reveals information about the achieved coverage. Another very important side effect of model construction is given by its deep analysis of specifications and/or requirements. We have developed systematic inspection techniques in which models are constructed systematically and at the same time requirements are analysed w.r.t. consistency and completeness. The presentation will focus on the following topics: 1. Sequence-based Inspection can be used to analyse requirements w.r.t consistency and completeness by constructing step by step a system model (or requirements model) by a well-defined process based on relevant input sequences and associated equivalence and output analysis. Each part of the model and each construction decision can be traced back to the requirements.
2. Statistical Testing draws a random sample from the input domain and exercises the system under test accordingly. It is usually based on a combined environment and system model, i.e. the system model, a Mealy Machine, is used to define a distribution on the input domain by adding transition probabilities (Markov Chain). Transition probabilities, weights, costs or criticality can than be used to select test cases. Model checker and standard mathematical algorithms can be used to validate the combined model, for example by proving safety properties of the Mealy Machine or by computing mean, variance or entropy information of the Markov Chain. Furthermore, a model checker can be used to generate test cases from the constructed models. A strategy for the complementary use of formal verification and testing could consist of (i) checking important properties of the constructed models, (ii) guaranteeing correctness of abstraction steps of the implementation by techniques like stepwise abstractions, (iii) guaranteeing correctness of refinement steps of the

system model, and (iv) generating test cases from important properties.

On Board Model Checking for Space Applications

Guiotto, A.^{1}; Roveri, M.^{2}; Cimatti, A.^{2}
^{1}Thales Alenia Space Italia; ^{2}Fondazione Bruno Kessler

In the framework of ESA research studies, Thales Alenia Space and Fondazione Bruno Kessler have investigated the use of model-checking in on-board space systems in order to increase their degree of autonomy.

In the traditional approach, the control of a spacecraft takes place mostly from ground, through the exchange of sequences of low level commands. Spacecraft is typically unable to deal alone with unexpected events from the environment or unpredicted on-board failures. In deep space and remote planetary exploration missions the limits in communication between ground and spacecraft (in time and bandwidth) increase reaction times and can decrease the efficiency of corrective actions.

Providing remote systems with the ability to create their own plans based on up-to-date information and enabling them to -re-plan in response to dynamic events would greatly improve the efficiency of a mission and potentially improve the safety of systems. Ground operators can use the restricted communication link to forward high-level mission objectives, which the on-board system can turn into detailed commands.

Execution can be monitored continuously and re-planning invoked when any execution problem occurred.

The software prototype developed in the study, called Autonomous Reasoning Engine (ARE), is based on model-based-reasoning. It is structured according to generic three-layer hybrid autonomy architecture: Deliberative, Executive and Control Layers. The Deliberative layer provides goal-driven planning and scheduling, plan validation and system-level FDIR facilities. The Executive Layer provides facilities to execute and monitor the correct execution of the current mission plan. The Control Layer provides low level interactions with the controlled system (sensor acquisition and commands to actuators sending).

The ARE relies on NUSMV, a symbolic model checker in the Deliberative and Executive Layers, where it performs all its reasoning on a formal model of the system it controls. The feedback control loop algorithms of the Control Layer are not based on symbolic reasoning, since complex numerical computations may be involved. The Control layer uses the model to encode low level sensor information, and to decode commands to be sent to actuators.

However, such computations are directly connected to the formal model through abstraction of the computation results into logical predicates. In this way, the computation steps are interleaved with logical reasoning at the higher levels. The formal model captures the intrinsic partial observability of the controlled system (available system sensors may not allow for conclusive determination of the controlled component's status). Development and validation of models is supported by a set of off-line model checking tools.

The approach is evaluated on two case studies (a planetary rover and an orbiting spacecraft), in order to characterize the approach in terms of reliability, availability and performances.

Requirements Modelling as an Effective Approach for Requirements Verification

Torget, Ø.; Tubaas, T.
Det Norske Veritas

DNV has in co-operation with other European companies defined a unified ISVV process for ESA, documented in the ESA ISVV-Guide. To validate this process in the context of real life projects, ESA awarded DNV a frame contract; "ISVV Process Validation". In Call-Off Order 2 (COO2) and Call-Off Order 3 (COO3) to this frame contract, DNV evaluated the use of modelling as an approach for performing requirements review.

Part of COO2 evaluated modelling requirements using the tool SpecTRM and the SpecTRM-RL modelling language. In SpecTRM models are a type of finite state machines with input, output, states and transitions. SpecTRM can automatically analyse the models for non-determinism and lack of robustness/completeness. COO3 evaluated both modelling of requirements in UML and model checking as a verification method. The model checking was focused on verification of requirements specified as UML diagrams. To perform the model checking the requirements were translated into UML diagrams and then model checked using the Maude model checker.

In both projects the modelling or translation of requirements into the modelling language was effective for finding unspecified behaviours in the specification and for increasing the reviewers understanding of the requirements. It is believed that the effectiveness to a large extent is due to the formalisation process that is required to perform modelling. Modelling requires that requirements are translated into the constructs of the modelling language. These constructs often have a precise and well-defined meaning and translating the requirements therefore forces the reviewer to look for the precise meaning of a requirement. Though there are several similarities, the approaches and tools used in the two projects have their strengths and weaknesses:
- SpecTRM can only model requirements that can be represented as finite state machines and this restriction makes it infeasible to model all parts of the requirements. The models can however be analysed for non-determinism and robustness/completeness which are relevant properties for state machines of models.
- UML is not a formal language and is in general not executable. This limits the type of automatic analysis

that can be performed. On the other hand UML is flexible and it has several different diagram types that can be used to model different types of requirements.
- The model checker Maude is very flexible so many different types of requirements can be modelled and formally analysed. The lack of a predefined way of translating requirements can however make the modelling expensive and difficult.
Both COO2 and COO3 indicate that modelling of requirements is an effective review technique, especially for discovering incomplete or missing requirements. Requirements modelling is however not good for discovering all types of issues and thus requirements modelling is suggested as an augmentation to the standard ISVV process for requirements verification rather than a replacement.

The ISVV Process Improvement - Rovsing A/S Position

Bundgaard, J.
Rovsing A/S

Improvement/revision to the ISVV process as defined in the ESA ISVV Guide.
The ISVV Guide could benefit from consolidated task descriptions and checklists. There are many additional stress testing techniques that could be listed in the guide. A catalogue of useful tools for automating parts of the analysis might save time. It might be beneficial to study which techniques are most likely to uncover problems that need fixing, and a study to determine the Return of Investment for ISVV (as it has been done at NASA). ISVV is supposed to create more "confidence" in the software product; maybe we need metrics to quantify this confidence - if possible at all?
Discussion on the usefulness of extending the Independent Verification process to earlier phases of the lifecycle.
The mantra has traditionally been "Earlier is better" for involvement of ISVV. Rovsing has recently participated in a review of an URD/Software System Specification and generated approx. 8% of the RIDs with a modest investment of time. Specifically, our findings mostly considered inconsistencies and lack of clarity. Domain knowledge is certainly required to be effective in this phase. A benefit to the ISVV supplier is that familiarisation with the software starts earlier. While this is an indisputably advantage it is more difficult to be quantitative about the value for money.

The use of Model Driven Engineering methods as a means of performing independent requirement baseline verification.
There is definitely a potential for improving the IVV outcome by independent construction of models on the basis of the requirements baseline - with the objective being the usual one to discover faults as well as undesired properties - and to do this as early as possible.

A model is a description of specific characteristics of a system at a high abstraction level. This allows the analyst to focus areas of particular interest, such as behavioural aspects (and to a lesser degree also structural and functional aspects) without the need for attention to implementation details and other low-level aspects. NASA is going down this route as far as verification is concerned.

The UML 2.0 behavioural modelling diagrams (use cases, activity-, state machine-, communication-, sequence-, and timing diagrams) should be considered as an obvious starting point, but other types of more formal models should also be considered.

Formal models are characterised by high degree of precision and complete internal consistence. The person who independently constructs the model is therefore likely to discover cases of over/under-specifications, contradictions, inconsistencies, omissions, lack of precision, unintelligible parts, and mere faults. However, the high precision of a formal model must be balanced against the relatively higher cost of preparing it compared to giving specifications in free text.

It is recommended that ISVV handbook is extended with processes for model-based requirements baseline verification.

ISVV Methodologies Exploration: ISVV focused on FDIR Verification

Silva, N.; Lopes, R.
Critical Software SA

The ESA Guide for Independent Software Verification and Validation is currently under review. The review aims to improve the processes already described in the Guide, re-organize them and also introduce new methodologies proved to be valuable in the ISVV activities. ISVV activities aim to improve software dependability. Software dependability is deeply connected to the ability of the software to recover from faults. In order to recover from faults, the software has to be able to detect them and be prepared to act in accordance in order to mitigate its impact. This ability is usually identified by FDIR (Fault Detection, Isolation and Recovery). In the scope of this task, we analysed the value of FDIR Verification for the ISVV activity. This analysis includes several steps:

• First, a verification of the current processes presented in ESA Guide for FDIR Verification.
• Second, an improvement of these processes in order to detail all the FDIR aspects to be verified.
• Finally, an evaluation of the results obtained by a FDIR Verification driven ISVV against the common ISVV process.

These steps were performed both for Requirements and for Design phase. This presentation includes the outputs of the several steps:

• First, the resultant Checklists as well as the process that lead to them.
• Second, the evaluation of the results obtained by applying the referred Checklists.

• Third, the evaluation of the Checklists (process validation), final conclusions and the proposed changes for the ESA ISVV Guide.

Cost Effective IV&V Planning based on Experiences from Spacecraft Projects

Matsumoto, T.; Kohtake, N.; Katahira, M.
Japan Aerospace Exploration Agency

Japan Aerospace Exploration Agency (JAXA) has implemented Software Independent Verification and Validation (IV&V) process to Spacecraft projects for several years. Recently the number of projects demanding Software IV&V has been increasing remarkably. In order to achieve cost effectiveness and higher quality in such circumstances, appropriate combinations of Software IV&V methods should be selected and applied to each project along certain guideline. In this presentation, we describe the approach for IV&V Planning based on Experiences from Spacecraft Projects. Then we will discuss the issues about that.

Until now, we applied IV&V Planning based on experiences and best guess gindvidually and manually h. As a result, significant defects were detected in the spacecraft software through IV&V activities. So, it was said to be effective. But, we think it can become more cost-effective by implementing IV&V Planning method more systematically (we call this gStrategic IV&V Planning h). For this purpose, we have established gIV&V guideline h to define standard of IV&V activities, and developed gStrategic IV&V Planning tool h to generate IV&V plans automatically along the IV&V guideline. But the tool has not become practical yet. One of the challenges of the Strategic IV&V Planning is effectiveness estimation of each IV&V activity. Now, we are addressing the issue by the approach of effectiveness measurement with defined metrics during IV&V activities executed in some Spacecraft Projects. In the future, we would like to collect the IV&V process data, analyze it, and feedback the result of the analysis to the IV&V Planning method to raise the accuracy of estimates.

In this presentation, we will discuss how to feedback the experience from Spececraft Projects to the following IV&V process effectively. We will especially focus on the effectiveness measurement.

A Galileo Perspective on ISVV

Latvala, T.
Space Systems Finland Ltd, FINLAND

Space Systems Finland Ltd (SSF) is the main provider of ISVV in the Galileo IOV phase. With a total of five ISVV contracts, with projects from the Ground Mission Segment and the Space Segment, SSF has gained unique experience in how Galileo ISVV projects function. Here we share some of the insights gained and make suggestions for improving the ESA ISVV process.

ISSV Review Process.Our five different contracts have not had a uniform ISVV review process, as customers have had different expectations on the process. This increased complexity, but it has given us the possibility to compare different review processes. The biggest differences between the processes are whether draft documentation is reviewed and should all findings be reported to the co-location meeting. In some cases the ISVV review has been allowed to continue some time after the co-location meeting. Common for all processes is that ISVV participates in the co-location meeting.

In all projects the practice of reviewing draft documentation was abandoned. Reviewing immature documents adds very little value. Although findings reported to the review meeting usually had the biggest impact, the post co-location review period is beneficial for plugging any perceived gaps in the ISVV review. ISVV participation in the review meetings is beneficial as it allows ISVV findings to be debated and also gives the ISVV contractor better insight into the software development.

Cost Drivers. Accurate estimates of costs are something that benefits both ISVV customers and providers. We have identified three generic cost drivers that impact the cost of performing ISVV:
• Software complexity: the more complex functionality implemented by the software, the more ISVV needs to spend on reviewing to ensure sufficient coverage of the material.
• Project complexity: if the software is split into several components that are reviewed separately and/or the project has a very unpredictable schedule, both project management and technical costs rise. A simple explanation is that it is more efficient to reserve a big chunk of resources on for one review effort than for several small ones. The impact of project complexity is comparable to software complexity.
• SVF maturity: preparing tests for a mature SVF environment is considerable less resource intensive than in the case where the SVF requires extensive adaptation. A mature SVF environment can offset the costs generated by greater software complexity.

Conclusions.Based on our analysis we think that the ESA ISVV guide should be updated to include more sophisticated cost models. This should give more realistic cost expectations for both ISVV customers and contractors. Currently there is only one model for the ISVV review process in the guide; explaining the benefits and disadvantages of different processes would make choosing the correct process for a project easier. A standard process should not be mandated in order to maintain flexibility.

ISVV Process for Autocoding

Bundgaard, J.
Rovsing

The presentation will address the essential output of the study "ISVV Process Definition for Auto-Code generated software". During this study the potential impacts of auto-code generation for the ISVV process have been analysed.

Introducing auto-code generation for mission critical on-board software is a promising technology, that automates some of the error-prone steps, which are present in the traditional manual software development process. When auto-code is used usually models are developed and consequently the efforts are moved from coding to specification and design level. The verification and validation of the generated auto-code remains necessary but must be adapted to the new process.

In this presentation the resulting amendments and extensions to the ISVV Guide, when auto-code is used, will be presented.

ISVV Guide - feedback to improve the ISVV Guide

Rodríguez-Dapena, P.
SoftWcare SL

As part of ESTEC Contract Nr: 18543/04/NL/LvH, SoftWcare SL is in charge of getting feedback from different projects and stakeholders about their experiences and proposed improvements to the currently existing Issue 1 of the ESA ISVV Guide. The inputs compiled to produce the new draft Issue 2 are from:

- Improved ISVV methods for efficiency and with empirical use of different methods for different verification activities executed on real projects eg: ATV, Aeolus, LisaPF
- Interviews with 20 different stakeholders: ISVV Customers, ISVV suppliers, SW suppliers, ESTEC and Ops team.
- Improvement of the Guide from known existing inconsistencies
- Improvement of the questionnaires in an Annex F of the ISVV Guide considering also for example IEEE 1012.

The main changes to the new draft Issue 2 of the ISVV Guide will be highlighted at the workshop including some controversial issues for which opposite inputs were collected.
Final Issue 2 of the ISVV guide is planned to be available by the End of 2008.