THE ASSERT SET OF TOOLS FOR ENGINEERING (TASTE): DEMONSTRATOR, HW/SW CODESIGN, AND FUTURE

Marc Pollina [(1)], Yann Leclerc [(1)], Eric Conquet [(2)], Maxime Perrotin[(2)], Guy Bois [(3)], Laurent Moss [(3)]

[(1)]M3Systems, 26, rue du soleil levant, 31410 Lavernose, France, Email: {pollina,leclerc}@m3systems.net
[(2)]ESA-ESTEC, Noordwijk, The Netherlands, Email: {eric.conquet, maxime.perrotin}@esa.int
[(3)] Space Codesign Systems Inc., 450 St-Pierre, suite 1010, Montreal, Quebec, H2Y 2M9 Canada, Email: {guy.bois, laurent.moss}@spacecodesign.com

ABSTRACT

This paper reports the results of a project funded by ESA on the use and development of **TASTE** (**T**he **A**SSERT **S**et of **T**ools for **E**ngineering). TASTE is a set of tools which, ruled by a clear methodology, aims to ease and secure the building of Real-Time Embedded (RTE) systems. The first goal of this project was to evaluate TASTE with an industrial case study, the realization of a satellite demonstrator, so as to confirm its maturity level. Technologies, design and application scenario of the demonstrator were chosen to be very realistic. The second goal of the project was to extend TASTE capabilities by adding hardware/software codesign support to the toolset. Many RTE systems are software and hardware, so adding such codesign support to TASTE broadens the range of targeted systems. For this purpose, basic hardware support was added to the toolset, and the combination between TASTE and SpaceStudio, a full HW/SW codesign environment, has been studied. With this project, we have been able to demonstrate that TASTE could be an answer to the "missing link" between high level system description and equipment design, for systems ranging from simple software-only systems to complex hardware/software systems.

Keywords
Embedded Systems, Real-time Systems, Hardware-Software Codesign, ASSERT, TASTE, SpaceStudio.

1. INTRODUCTION

For critical systems, improving the efficiency of the engineering process while increasing the functional complexity is a well known challenge and it is one of the keys for competitiveness of space industry in the near future.

As you can expect this challenge is very difficult to tackle. It means many things at the same time and it encompasses issues related to companies' organisations, maturity of tools and technologies but also cultural "convictions" of all stakeholders.

We face this challenge by recognising these difficulties but also by being convinced that progress is possible and that step by step industrial implementation of innovative model based engineering process is feasible today.

This conviction comes from the methodological and technical results obtained in the frame of the **TASTE** (**T**he **A**SSERT **S**et of **T**ools for **E**ngineering) project. We will present in this paper this methodological framework, the application case considered for the project, and finally the options that enable the TASTE engineering process including HW/SW co-design.

2. TASTE PROJECT

2.1. Context

The history of our project starts with **ASSERT** (**A**utomated proof-based **S**ystem and **S**oftware **E**ngineering for **R**eal-**T**ime systems), an Integrated Project partially funded by the European Commission from 2004 to 2007 [1]. In those days ASSERT was focused on software engineering with the ultimate goal of supporting design and verification of

complex and critical applications. The objectives of ASSERT were to develop a reliable and scientific approach for software and system engineering. From 2008 to early 2010, ESA completed the work initiated in ASSERT. This led to the release of successive versions of TASTE, the toolset supporting the ASSERT process, up to an advanced prototype of the tools technology.

The TASTE project, funded by ESA, is the continuation of this work. It was managed by M3Systems in partnership with academics (ISAE), actors from space industries (EADS Astrium, Thales Alenia Space), and specialized SMEs (SEMANTIX, Space Codesign Systems, LVDH).

2.2. TASTE

The purpose of TASTE [2, 3] is to build Real-Time Embedded (RTE) systems that are correct by construction: based on high-level models, the toolset automatically configures and deploys complex RTE systems. To do so, TASTE relies on key standards and technologies such as:

- The Ravenscar Computational Model (RCM). The computational model enforced by the generated system, stemming from the Ravenscar profile [4].
- ASN.1 [5] and AADL [6] for systems modeling.
- Code/Model skeleton generators targeting major programming languages (C, C++, Ada) and modeling tools (SCADE, Simulink…) to help in designing systems' functionalities.
- Code generators to automatically generate systems, based on high-level models.
- PolyORB-HI. A middleware to map the primitives of generated codes to the ones offered by targeted operating systems.

At the beginning of this study, TASTE-generated systems were software only, and were targeting multiple platforms: x86 (with Linux, Mac OS X, FreeBSD, RTEMS), ARM (with RTEMS, Linux), SPARC/LEON (with RTEMS or OpenRavenscar).

2.3. Objectives of the project

With this project, our aim was to go one step further in TASTE assessment and development: by exposing the technology to industrial experts and real cases, so as to confirm its maturity level, and by opening the way for new concepts to be integrated into the proposed approach and toolset.

More precisely, the project focused on:

- The realization of a case study representative of a satellite demonstrator
- The extension of the toolset by integrating basic hardware support, and by studying the relevance and feasibility of the combination of TASTE and SpaceStudio, a cutting edge HW/SW codesign environment [7, 8].

3. TASTE DEMONSTRATOR

The TASTE demonstrator has been designed with the goal to maximize case study's realism: Satellite demonstrator had to perform "look alike" satellite functions, be based on common space technology, and be part of a realistic application scenario.

Such realism is essential to get the most of the case study when assessing the toolset.

3.1. Application scenario

The satellite demonstrator is aimed to provide a TM/TC (Telemetry/Telecommand) encryption service working on geographical windows, as a function of its position (see Fig.1).

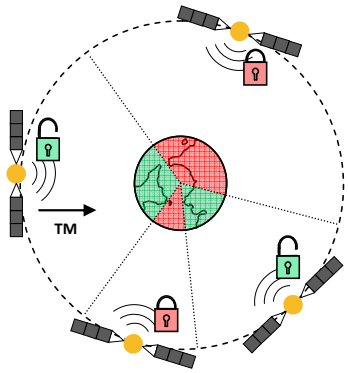

Figure 1 Application Scenario

In other words, the satellite sends encrypted or unencrypted TMs to the ground, depending on earth's area flown over. Encryption areas and other satellite settings are managed from the ground through TCs.

This application scenario is a rough but realistic representation of a typical satellite mission.

3.2. Demonstrator definition

So as to perform the described scenario, the satellite demonstrator has been divided between a payload and a platform (see Fig. 2):

- The payload is a GPS receiver, computing the satellite's position from GPS signals, and providing the results to the platform.
- The platform performs the TM/TC encryption service: it computes the geographical area flown over by the satellite and any encryption of the TMs depending on the result. In addition, the platform processes the incoming TCs.

This division between payload and platform is common in space industry. It was a good way to share the design of the demonstrator between teams, regarding their fields of expertise.

As shown in Fig. 2, technologies involved in demonstrator development have been chosen amongst typical space ones: LEON processors, Spacewire links, etc...

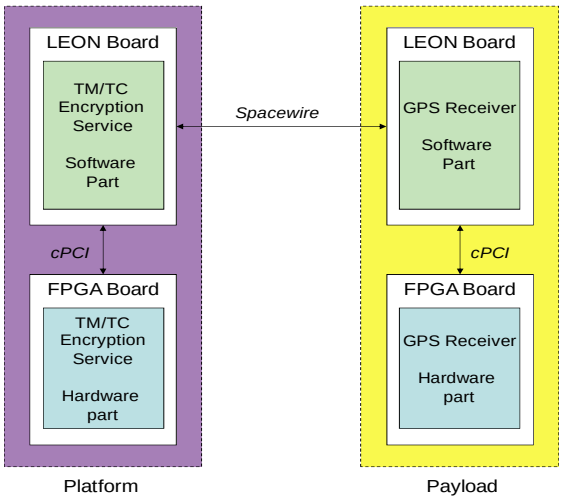

Figure 2 Demonstrator architecture

In addition, ground stations are simulated with 2 PCs linked to the demonstrator.

3.3. Lessons learnt

The relevance of TASTE for the design of complex systems (distributed, heterogeneous, critical, and mixed hardware/software) has been highlighted during the development of the demonstrator. The use of recognized standards (ASN.1, AADL, RCM), and a strong automation of the design process, are key points of the approach.

The TASTE approach is helpful in designing a system as it allows sharing the development between “specialized” teams, with different development languages/tools, while simplifying integration and enforcing specific constraints (RCM constraints).

The newly added support for hardware components (cf. §4) in system designs was the opportunity to benefit from powerful hardware components as part of the TASTE demonstrator. This feature is a key prerequisite for the design of mixed HW/SW embedded systems.

TASTE in its current form is useful to experiment on principles and gain feedback on feasibility/complexity/interest of techniques and methods. It can support the definition of innovative concepts and provides a prototyping framework for R&D activities.

In order to reach industry expectations for use in an operational project, evolution is still required; introducing a clear separation between engineering/design phases, improving the user support, and increasing the role of physical architecture modelling.

4. TASTE TOOLSET AND HW/SW CODESIGN

Both TASTE’s own hardware/software codesign capabilities and interfacing with cutting-edge codesign tools have been experimented with during this project.

4.1. Own codesign capabilities

Event though TASTE has primarily been designed to cope with software systems, the ASSERT methodology and the overall approach of TASTE are very general, and can easily be extended to mixed hardware-software systems.

Therefore, by following TASTE’s and ASSERT methodology’s requirements, basic codesign capabilities were added to the toolset. This mainly consisted of:

- Adding a support for SystemC- [9] and VHDL-based design of simple hardware functions to the toolset.
- Extending the list of targeted devices, by making FPGA components available to TASTE users.

With these improvements in the toolset, mixed hardware-software systems design was made as easy as purely software ones.

4.2. SpaceStudio/TASTE integration study

The combination of a cutting edge HW/SW codesign tool, SpaceStudio from Space Codesign Systems [7, 8], and TASTE, was examined during the study. Goals were to assess the advantages and benefits of powerful codesign capabilities in complex systems development with TASTE, and to pave the way for an efficient integration of codesign capabilities into the toolset.

Following the evaluation of both TASTE and SpaceStudio, we have been able to establish the strong points of each toolset. In short, TASTE has been recognized for its strengths in the specification of complex distributed systems' functionalities. On the other hand, SpaceStudio's advanced capabilities for the design, analysis and implementation of complex Systems on Chip (SoC) have been highlighted. The rest of this section introduces SpaceStudio, presents a case study based on implementing a TASTE function as a SpaceStudio-modelled HW/SW SoC, and presents further integration possibilities.

4.2.1. SpaceStudio overview

SpaceStudio is an integrated development environment and tool suite providing a complete HW/SW co-design flow and platform, with the ability to transform functions between HW and SW without recoding as designers decide on the makeup of their system. The four major components of SpaceStudio are presented in Fig. 3: Elix, Simtek and GenX allow for design at three different levels of abstraction (functional, architectural, implementation), while SpaceMonitor provides performance analysis and profiling.

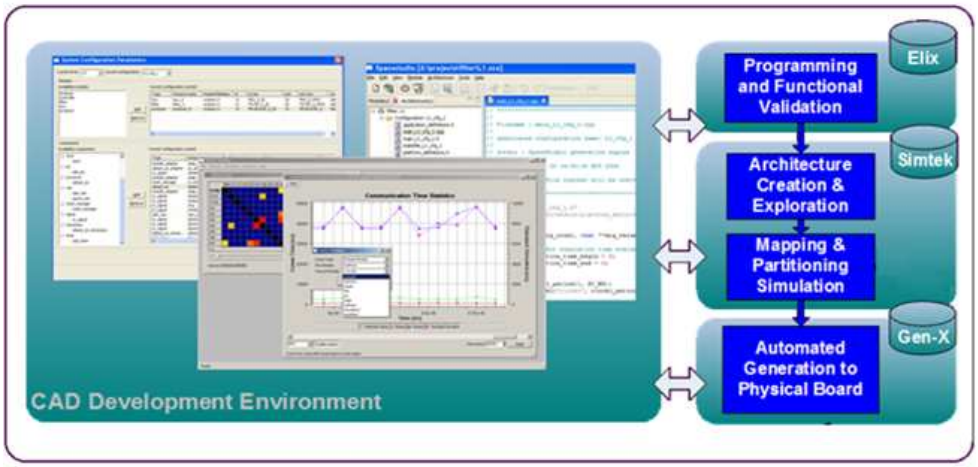

Figure 3 SpaceStudio tool suite

First, functional specification is supported by SpaceStudio's Elix, working at a high level of abstraction using component libraries and user-supplied C/C++/SystemC blocks. Communication between blocks can be specified using several mechanisms and semantics (e.g., message passing, shared-memory, etc.). Elix also enables functional validation of specifications by generating executable models of the system to perform untimed or timed functional simulations.

Next, the system's architecture is created in SpaceStudio's Simtek, using component libraries and the mapping of Elix-validated functional blocks to the architecture's components. This powers high-level architectural exploration with

SpaceStudio to test different system architectures (e.g., number of processors and busses, task mapping on HW or SW, etc.). Simtek then automatically generates a SystemC virtual platform and C/C++ embedded SW, modeling the given functionality, communications, HW/SW architecture and mapping. This enables fast HW/SW co-simulation and co-debugging of the system before committing to a particular FPGA or ASIC implementation.

SpaceMonitor offers seamless and non-intrusive profiling [10] of Elix executable models and Simtek virtual platforms, extracting performance metrics on both HW- and SW-mapped functional blocks, such as: bus and memory utilization, processor load and RTOS scheduling and context switches. This information can be used to make informed decisions with respect to Elix-based functional specification and Simtek-based high-level system architecture.

For implementation of the HW portion of the system, SpaceStudio's GenX (based on IP-XACT [11]) generates and integrates the required HW IPs, glue logic, firmware, embedded SW and configuration files to implement the Simtek-specified system design into the final target (e.g., FPGA) using a standard downstream RTL flow. GenX can also support HLS flows from C/C++ and SystemC to generate and integrate application-specific HW RTL blocks [12].

4.2.2. SpaceStudio/TASTE case study

The first integration possibility that has been considered is to support implementing a TASTE function as a SpaceStudio-modeled SoC. The aim of this approach is to provide a way to benefit from the strength of SpaceStudio when designing a TASTE system, without major changes to TASTE or SpaceStudio. As shown in Fig. 4, the main idea is to use SpaceStudio like any of TASTE's modeling tools (Simulink, SCADE…).

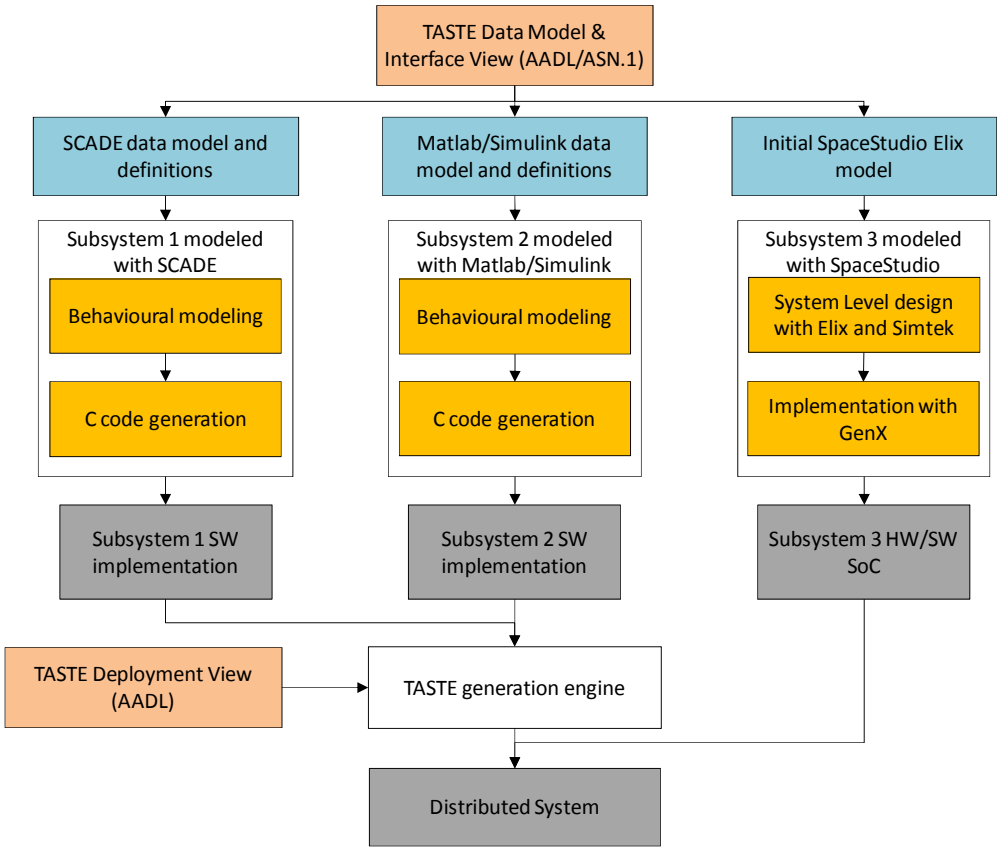

Figure 4 TASTE function as a SpaceStudio-modeled SoC – Design flow

Thus each TASTE function targeting SpaceStudio as a modelling tool is designed through SpaceStudio as an SoC and implemented in a way that allows interfacing it seamlessly with the standard TASTE-generated part of the system. The designer is then free to follow the standard SpaceStudio flow presented in §4.2.1.

A simplified prototype implementation has been tested to assess the relevance of this approach. It relies on the TASTE concept of blackbox, which is a peripheral device providing a service to a TASTE system through a specific driver. Such a blackbox is used to run a SpaceStudio SoC driver, allowing TASTE functions to interface with a SpaceStudio-modeled SoC. Thus, we have been able to design part of a TASTE system, the Channel 1 part of the TASTE demonstrator's GPS receiver, as an SoC within SpaceStudio. We then co-simulated it with TASTE's native execution using a TCP socket as a high-level communication mechanism, as shown in Fig. 5. This is a first proof of concept of taking advantage of SpaceStudio's design space exploration and performance monitoring features in a TASTE project.

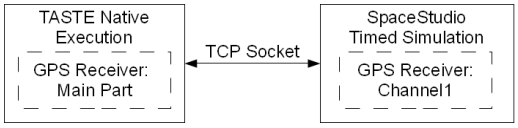

Figure 5 Overview of connection between TASTE execution and SpaceStudio simulation

The first step was to create an Elix functional model of the Channel 1 part. This models two different aspects, as shown in Fig. 6: 1) communications with the rest of the GPS receiver; and 2) the core functionality of the Channel 1 part.

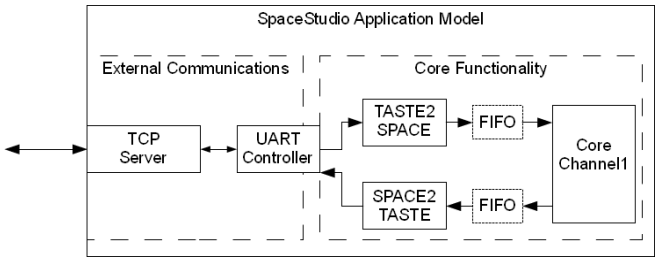

Figure 6 Overview of SpaceStudio application modelling of the Channel 1 part

SpaceStudio's libraries contain several data source and sink components that can inject inputs into a simulation and extract outputs from it. Here, we used a TCP server component, which communicates with the TASTE native execution and binds the socket's data stream to a SystemC TLM-2.0 [13] model of an I/O peripheral within the SpaceStudio simulation, in our case an UART controller model from SpaceStudio's library. Such component is useful as it allows early high-level functional validation of the SpaceStudio-modelled application's interfacing with the rest of the system without requiring deployment and configuration of the actual interface.

The core functionality of the Channel 1 part consists of receiving channel configuration commands, processing each command, and sending back channel status. To this end, three SpaceStudio application modules have been modelled: the TASTE2SPACE and SPACE2TASTE Helpers, and the Core Channel 1. The Helpers are intermediate between the inner SoC and the TASTE native execution: they send/receive data through the UART interface, perform ASN.1 encoding/decoding, and communicate with the Core Channel 1 via FIFO-based message passing. The core Channel 1 reads and processes the input command, computes the new channel status and outputs the result.

This SpaceStudio application modelling was validated by performing Elix functional simulations of the Channel 1 part of the GPS receiver together with a TASTE native execution of the main part, which confirmed its correct behaviour.

Once modelled within SpaceStudio, different HW/SW architectures have been created for the Channel 1 and several mapping options have been tested. For the main tests, the TASTE2SPACE and SPACE2TASTE modules have been mapped into HW, as they would probably be mapped into HW on the realized SoC. The following architectures and mappings have therefore been tested: all-HW architecture (no processor and Channel 1 application module mapped into HW), MicroBlaze-based architecture (Channel 1 application module mapped onto a Xilinx MicroBlaze), LEON-based architecture (Channel 1 application module mapped onto a LEON3, with or without caches enabled), and ARM-based architecture (Channel 1 application module mapped onto an ARM Cortex-A9).

For each architecture and mapping, SpaceStudio was used to generate and build a SystemC TLM-2.0 virtual platform including the HW-mapped application modules and platform models as well as the connections between those HW components. Where applicable, SpaceStudio was also used to generate for each processor ANSI C/C++ embedded SW, including the SW-mapped application modules, RTOS, as well as the BSP and drivers, all built into an ELF executable image. SpaceStudio was then used to simulate the generated embedded SW on the TLM virtual platform, representing a full simulation of both the HW and SW parts of the subsystem. All the architectures have been confirmed to function properly, and performance monitoring and analysis were applied to the case study. For instance, we profiled the execution time of SW tasks on the LEON processor. The breakdown of processor usage between the core Channel 1 task, interrupt service routines, context switching and idle time is shown in Fig. 7 with caches enabled and disabled.

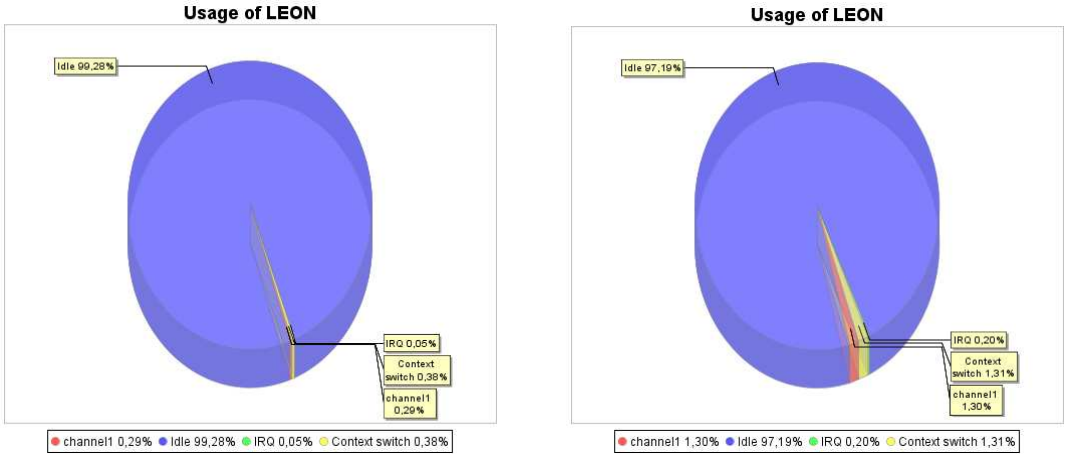

Figure 7 Processor usage with instruction and data caches enabled (left) or disabled (right)

We see that enabling caches significantly decreases processor usage in this case. Also, the fact that processor usage is at around 1-3% is useful information for system architects or embedded SW designers, it indicates that additional application modules or system functionality could be mapped onto that processor.

After selection of the HW/SW architecture and mapping for the SoC, the next step would be to create a HW/SW FPGA SoC implementation through GenX, and to integrate it into the overall realized TASTE system. This step was outside of the scope of this study.

4.2.3. Further TASTE/SpaceStudio integration possibilities

Further integration possibilities between SpaceStudio and TASTE have been explored during the study. These possibilities are summarized here.

SpaceStudio as a virtual platform for TASTE systems: This approach uses SpaceStudio as a virtual platform, or a set of virtual platforms, so as to run TASTE generated SW binaries on it. The result of the simulations is then used to assess the behaviour and the performance of the designed systems, and modify their models when relevant.

Distribution of TASTE systems over SpaceStudio-modelled SoCs: This approach gives a designer the ability to map the subsystems of a TASTE system, not only a component, to a SpaceStudio-modelled SoC through an automated process. Thus, a SpaceStudio-modeled SoC plays the same role as a TASTE processor board.

SpaceStudio-driven refinement of TASTE deployment view: This approach uses SpaceStudio to refine a functional TASTE specification (Data View + Interface View) into a TASTE architectural specification (Deployment View). After

this design space exploration, the TASTE toolset would handle the implementation of individual functions and the implementation generation for the SpaceStudio-selected architecture

5. CONCLUSION AND FUTURE

The ASSERT methodology and its evolution with TASTE have been developed in different phases since 2003. Initially, an FP6 project co-financed by the European Commission has set the picture and decided a focus. Complementary work has been financed by ESA, until now, to improve key features, leading to successive versions of the toolset.

The relevance of TASTE for the design of complex embedded systems has been highlighted during this study, with the development of the TASTE demonstrator. The toolset in its current version is useful to support the definition of innovative concepts and provides a prototyping framework for R&D activities.

In order to progress in the industrial direction, a step by step evolution is required, starting with those capabilities that are the most interesting for industry (easy to adopt and with tangible benefits). Bridges between TASTE and other related studies (e.g. SAVOIR-FAIRE [14]) would be essential to help industrial adoption of the approach.

Preparing for the future also means supporting innovative concepts, such as HW/SW codesign. The result of this study, with respect to the combination of TASTE with SpaceStudio, is the first step in that direction.

Obviously these steps require investment on the users' side: implementing a new technology is not free. In this industrial challenge, innovation clusters and space agencies can support industry in this essential and necessary evolution, in order to maintain a high level of competitiveness in European Space industry.

REFERENCES

[1] Conquet, E. "The assert-project: a step towards a reliable and scientific system and software engineering". ERTS'2008.Toulouse, France, January 2008

[2] Perrotin, M., Conquet, E., Delange, J., Schiele, A., Tsiodras, T.: TASTE: A Real-Time Software Engineering Tool-Chain, Oveview, Status and Future, *Proceeding of the 15th International SDL Forum*. Toulouse, France, July 2011

[3] Perrotin, M., Conquet, E., Dissaux, P., Tsiodras, T., Hugues, J.: "The TASTE Toolset: turning human designed heterogeneous systems into computer built homogeneous software". ERTS'2010, Toulouse, France, 2010

[4] Burns, A., Dobbing B., Vardanega, T. 2004. Guide for the use of the Ada Ravenscar Profile in high integrity systems. *Ada Letters.* XXIV, 2 (June 2004), pp. 1-74.

[5] Larmouth J. *ASN.1 Complete*. Elsevier, 2000

[6] Feiler, P., Gluch, D. Hudak, J. *The Architecture Analysis & Design Language (AADL): An Introduction*. Carnegie Mellon University, 2006

[7] Bois G., Moss L., Filion L., Fontaine S. "Codesign Experiences Based on a Virtual Platform," in *ESL Models and their Application: Electronic System Level Design and Verification in Practice*, Brian Bailey and Grant Martin (Eds.) New York, NY: Springer, 2010, pp. 273-308.

[8] Space Codesign System Inc. Product Highlights. Online at http://www.spacecodesign.com/ (as of December 7, 2011).

[9] IEEE Std 1666-2005. *IEEE Standard SystemC Language Reference Manual*.

[10] Moss L., de Nanclas M., Filion L, Fontaine S., Bois G., Aboulhamid M. “Seamless Hardware/Software Performance Co-Monitoring in a Codesign Simulation Environment with RTOS Support.” *Proceedings of the 10th Design, Automation and Test in Europe Conference and Exhibition* (DATE'07), pp. 876-881.

[11] IEEE Std 1685-2009. *IEEE Standard for IP-XACT, Standard Structure for Packaging, Integrating, and Reusing IP within Tool Flows*.

[12] Moss L., Cantin M.-A., Bois G., Aboulhamid M. “Automation of Communication Refinement and Hardware Synthesis within a System-Level Design Methodology”, *Proceedings of the 19th IEEE/IFIP International Symposium on Rapid System Prototyping* (RSP '08), IEEE Computer Society, Washington, DC, USA, 75-81 – 2008

[13] IEEE Std 1666-2011. *IEEE Draft Standard for Standard SystemC Language Reference Manual*.

[14] Savoir-Faire working group. “Space On-board Software Reference Architecture”. DASIA2010. 2010