Project DEPLOY
Grant Agreement 214158
"Industrial deployment of advanced system engineering methods for high productivity and dependability"

D39 D3.2 – Report on Enhanced Deployment in the Space Sector

5.8.2011

http://www.DEPLOY-project.eu

Contributors to This Document

SSF:

Dubravka Ilić
Timo Latvala
Laura Nummila
Tuomas Räsänen
Pauli Väisänen
Kimmo Varpaaniemi

Åbo Akademi University:

Linas Laibinis
Yuliya Prokhorova
Elena Troubitsyna

University of Newcastle upon Tyne:

Alexei Iliasov
Alexander Romanovsky

University of Southampton:

Michael J. Butler
Asieh Salehi Fathabadi
Abdolbaghi Rezazadeh

CETIC:

Jean-Christophe Deprez
Renaud De Landtsheer
Christophe Ponsard

Reviewers of This Document

Kaisa Sere (Åbo Akademi University)
Colin F. Snook (University of Southampton)

Table of Contents

1 Introduction

1.1 Aims of the Work

The purpose of this deliverable is to report on the enhanced deployment activities in WP3. The main goal of SSF's involvement in the DEPLOY project is integrating the Rodin Platform integration chain with respect to the current SSF processes. This complex goal is reflected in several more focused sub-goals investigated in the two conducted pilots. Summarized below are the aims of the work carried out so far:

- Requirements engineering—investigate the support of the project tools and methods in improving requirements elicitation and management.
- Modeling approach—abstractions, managing complexity. In particular, using modular and decompositional Event-B extensions in order to avoid unnecessary work in preparation and maintenance of models and proofs.
- Suitability—analyze (sub)systems particularly suitable to be modeled with Event-B / evaluate suitability of Rodin Platform and its plug-ins.
- Getting to know how to model typical architectures.
- Application of Event-B to designs with limited prior knowledge about good solutions to associated design problems.
- Using Event-B in order to support RAMS (Reliability, Availability, Maintainability and Safety) activities.
- Finding ways to deal with real-time properties in Event-B.

1.2 Document Structure

This document is structured in the following way:

- Section 2 summarizes the WP3 work reported in previous DEPLOY deliverables.
- Section 3 summarises the status of enhanced deployment.
- Section 4 reports on the different activites performed for the enhanced deployment.

Section 4.1 uses DSAOCSS (Distributed System for Attitude and Orbit Control for a Single Spacecraft) as an example of how Event-B can be applied to a design with limited prior knowledge about good solutions to associated design problems.

Section 4.2 reports the WP3 Event-B research done on real-time.

Section 4.3 considers derivation of behavioural specifications from FMEA (Failure Modes and Effects Analysis).

Section 4.4 reports the WP3 experience of using modular Event-B.

Section 4.5 reports the WP3 experience of using decompositional Event-B.

Section 4.6 describes activities in and experience from SSF's internal Event-B training.

- Section 5 summarizes the evidence collected on enhanced deployment in WP3.
- Section 6 makes recommendations based on lessons learnt during enhanced deployment and presents potential further activities.
- Section 7 draws overall conclusions from enhanced deployment in WP3.

1.3 Abbreviations

AOC	Attitude and Orbit Control
AOCS	Attitude and Orbit Control System
ASW	Application Software
BC	BepiColombo
CD	Compact Disc
CETIC	Centre d'Excellence en Technologie de l'Information et de la Communication
CPU	Central Processing Unit
CSW	Core Software
DAOCS	Decentralised AOCS
DHSW	Data Handling Software

DPU	Data Processing Unit
DSAOCSS	Distributed System for Attitude and Orbit Control for a Single Spacecraft
ECSS	European Cooperation for Space Standardization
e.g.	exempli gratia
ES	Earth Sensor
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre
Ed.	Editor
Eds.	Editors
etc.	et cetera
FMEA	Failure Modes and Effects Analysis
FPA	Focal Plane Assembly
FPGA	Field-Programmable Gate Array
FDIR	Failure Detection, Isolation and Recovery
GPS	Global Positioning System
HW	Hardware
ID	Identifier
i.e.	id est
LNCS	Lecture Notes in Computer Science
MIXS	Mercury Imaging X-ray Spectrometer
MIXS-C	MIXS Collimator
MIXS-T	MIXS Telescope
NASA	National Aeronautics and Space Administration
OBSW	On-Board Software
PLC	Programmable Logic Controller
PLI	Payload Instrument

PSU	Power Supply Unit
PUS	Packet Utilization Standard
pp.	pages
QA	Quality Assurance
RAMS	Reliability, Availability, Maintainability and Safety
ROM	Read-Only Memory
RTOS	Real-Time Operating System
RW	Reaction Wheel
SIXS	Solar Intensity X-ray and Particle Spectrometer
SIXS-P	SIXS Particle Detector Part
SIXS-X	SIXS X-Ray Detector Part
SRD	Software Requirements Document
SS	Sun Sensor
SSF	Space Systems Finland, Ltd.
STR	Star Tracker
SW	Software
TC	Telecommand
THR	Thruster
TM	Telemetry
WP	Work Package
w.r.t.	with respect to

1.4 References

1.4.1 Applicable Documents

[AD1] Information Society Commission of the European Communities, Information Media Directorate-General, and Communication Technologies. *Seventh Framework Programme Grant Agreement No 214158, Industrial Deployment of Advanced System Engineering Methods for High Productivity and Dependability. Collaborative Project*. December 2007.

[AD2] Grant Agreement [AD1] *Annex I – Description of Work*. Approved by EC in November 2007.

[AD3] DEPLOY Project. *Industrial Deployment of Advanced System Engineering Methods for High Productivity and Dependability. Consortium Agreement*. May 2008.

[AD4] DEPLOY Project. *DEPLOY Deliverable D18 – D12.3 Project Refocus – Amended Description of Work*. August 2009.

1.4.2 Reference Documents

[RD1] ECSS Secretariat, ESA-ESTEC, Requirements & Standards Division. *Space Engineering: Ground Systems and Operations – Telemetry and Telecommand Packet Utilization (ECSS-E-70-41A)*. http://www.ecss.nl/, January 2003.

[RD2] ESA Media Center, Space Science. *Factsheet: BepiColombo*. http://www.esa.int/esaSC/SEMNEM3MDAF_0_spk.html, January 2008.

[RD3] DEPLOY Project. *DEPLOY Deliverable D5 JD1: Report on Knowledge Transfer*. http://www.deploy-project.eu/pdf/fv-d5-jd1-reportonknowledgetransfer.zip, January 2009.

[RD4] DEPLOY Project. *DEPLOY Deliverable D7 D11.1: Measurement Methodology Guide*. http://www.deploy-project.eu/pdf/d7-revised-final.pdf, August 2009.

[RD5] DEPLOY Project. *DEPLOY Deliverable D20 D3.1 – Report on Pilot Deployment in the Space Sector*. http://www.deploy-project.eu/pdf/D20-pilot-deployment-in-the-space-sector-final-version.pdf, January 2010.

[RD6] DEPLOY Project. *DEPLOY Deliverable D29 (JD2) – Initial Assessment Results*. http://www.deploy-project.eu/pdf/D29-JD2-V1.0.pdf, September 2010.

[RD7] Event-B and Rodin Platform Documentation Wiki. *Rodin Plug-ins*. http://wiki.event-b.org/index.php/Rodin_Plug-ins, June 2011.

[RD8] Asieh Salehi Fathabadi, Abdolbaghi Rezazadeh, and Michael J. Butler. *Event-B Project BepiColombo_Soton_v18.0 (Rodin Archive of Space System)*. http://eprints.ecs.soton.ac.uk/22048/, October 2010.

[RD9] Asieh Salehi Fathabadi, Abdolbaghi Rezazadeh, and Michael J. Butler. *Applying Atomicity and Model Decomposition to a Space Craft System in Event-B*. In Mihaela Bobaru, Klaus Havelund, Gerard J. Holzmann, and Rajeev Joshi (Eds.), *NASA Formal Methods, Third International Symposium, NFM 2011, Pasadena, CA, USA, April 18–20, 2011, Proceedings*, LNCS 6617, pp. 328–342, Springer-Verlag, http://www.springerlink.com/content/d8q6028518110157/ and http://eprints.ecs.soton.ac.uk/22048/, April 2011.

[RD10] Alexei Iliasov. *A Lecture on Modularisation Method and Plug-in: Introduction*

and Parking Lot Case Study. http://deploy-eprints.ecs.soton.ac.uk/227/, May 2010.

[RD11] Alexei Iliasov. *Tutorial on the Modularisation Plug-in for Event-B*. In *Workshop on B Dissemination, Natal, Brazil, November 8–9, 2010*. http://deploy-eprints.ecs.soton.ac.uk/263/, December 2010.

[RD12] Alexei Iliasov, Linas Laibinis, and Elena Troubitsyna. *An Event-B Model of the Attitude and Orbit Control System*. http://deploy-eprints.ecs.soton.ac.uk/213/, March 2010.

[RD13] Alexei Iliasov, Linas Laibinis, Elena Troubitsyna, Alexander Romanovsky, and Timo Latvala. *Augmenting Event B Modelling with Real-Time Verification*. Technical Report 1006, Turku Centre for Computer Science, http://tucs.fi/research/publication-view/?pub_id=tIlLaTrRoLa11a, April 2011.

[RD14] Alexei Iliasov, Elena Troubitsyna, Linas Laibinis, Alexander Romanovsky, Kimmo Varpaaniemi, Dubravka Ilić, and Timo Latvala. *Supporting Reuse in Event B Development: Modularisation Approach*. In Marc Frappier, Uwe Glässer, Sarfraz Khurshid, Régine Laleau, and Steve Reeves (Eds.), *Abstract State Machines, Alloy, B and Z: Second International Conference, ABZ 2010, Orford, Québec, Canada, February 22–25, 2010, Proceedings*, LNCS 5977, pp. 174–188, Springer-Verlag, http://www.springerlink.com/content/1246876j83576368/, February 2010.

[RD15] Alexei Iliasov, Elena Troubitsyna, Linas Laibinis, Alexander Romanovsky, Kimmo Varpaaniemi, Dubravka Ilić, and Timo Latvala. *Developing Mode-Rich Satellite Software by Refinement in Event B*. Technical Report CS-TR-1207, School of Computing Science, University of Newcastle upon Tyne, http://www.cs.ncl.ac.uk/publications/trs/abstract/1207, June 2010.

[RD16] Alexei Iliasov, Elena Troubitsyna, Linas Laibinis, Alexander Romanovsky, Kimmo Varpaaniemi, Dubravka Ilić, and Timo Latvala. *Developing Mode-Rich Satellite Software by Refinement in Event B*. In Stefan Kowalewski and Marco Roveri (Eds.), *Formal Methods for Industrial Critical Systems: 15th International Workshop, FMICS 2010, Antwerp, Belgium, September 20–21, 2010, Proceedings*, LNCS 6371, pp. 50–66, Springer-Verlag, http://www.springerlink.com/content/w8281922t2471515/, September 2010.

[RD17] Alexei Iliasov, Elena Troubitsyna, Linas Laibinis, Alexander Romanovsky, Kimmo Varpaaniemi, Pauli Väisänen, Dubravka Ilić, and Timo Latvala. *Verifying Mode Consistency for On-Board Satellite Software*. Technical Report 971, Turku Centre for Computer Science, http://tucs.fi/research/publication-view/?pub_id=tIlTrLaRoVaVaIlLa10a, April 2010.

[RD18] Alexei Iliasov, Elena Troubitsyna, Linas Laibinis, Alexander Romanovsky, Kimmo Varpaaniemi, Pauli Väisänen, Dubravka Ilić, and Timo Latvala. *Verifying Mode Consistency for On-Board Satellite Software*. In Erwin Schoitsch (Ed.), *Computer Safety, Reliability, and Security: 29th International Conference, SAFECOMP 2010, Vienna, Austria, September 14–17, 2010,*

Proceedings, LNCS 6351, pp. 126–141, Springer-Verlag, http://www.springerlink.com/content/7hv48n2h38128363/, September 2010.

[RD19] Dubravka Ilić, Timo Latvala, Kimmo Varpaaniemi, Pauli Väisänen, Elena Troubitsyna, and Linas Laibinis. *Evaluating a Control System Architecture Based on a Formally Derived AOCS Model*. In *DASIA 2010: DAta Systems In Aerospace, Budapest, Hungary, June 1–4, 2010, Proceedings*, ESA SP-682 (CD-ROM), European Space Agency, June 2010.

[RD20] Yuliya Prokhorova, Elena Troubitsyna, Linas Laibinis, Kimmo Varpaaniemi, and Timo Latvala. *Deriving Mode Logic for Fault-Tolerant Control Systems*. In *Proceedings of NODES 2011, 5th Nordic Workshop on Dependability and Security, Copenhagen, June 27–28, 2011*, Technical Report of Technical University of Denmark, June 2011.

[RD21] Tuomas Räsänen and Laura Nummila. *DEPLOY Training Evaluation Document*. http://deploy-eprints.ecs.soton.ac.uk/314/, September 2010.

[RD22] Kimmo Varpaaniemi. Rodin Platform bug tracker items 3053410, 3056815, 3058915, 3061858, 3066966, 3067672, 3073433, 3074011, 3084683, 3088874, 3089203, 3093913, 3116400, 3134438, 3143114, 3150302, 3150394, 3151189, 3151805, 3154319 and 3206302, http://sourceforge.net/tracker/?atid=651669&group_id=108850&func=browse, August 2010 – March 2011.

[RD23] Kimmo Varpaaniemi. Rodin Platform feature request tracker items 3052182, 3067513, 3067580, 3152959 and 3155514, http://sourceforge.net/tracker/?atid=651672&group_id=108850&func=browse, August 2010 – January 2011.

[RD24] Kimmo Varpaaniemi. *Event-B Project BepiColombo_Models_v6.4*. http://deploy-eprints.ecs.soton.ac.uk/244/, September 2010.

[RD25] Kimmo Varpaaniemi. *DEPLOY Work Package 3 Attitude and Orbit Control System Software Requirements Document*. DEP-RP-SSF-R-005, Issue 1.0, http://deploy-eprints.ecs.soton.ac.uk/266/, December 2010.

[RD26] Kimmo Varpaaniemi. ProB defect report tickets #96 and #102, http://cobra.cs.uni-duesseldorf.de/trac/report (where the "All Tickets" view is actually not guaranteed to cover the "Active Tickets" view), March 2011.

[RD27] Kimmo Varpaaniemi. *DEPLOY Work Package 3 Software Requirements Document for a Distributed System for Attitude and Orbit Control for a Single Spacecraft*. DEP-RP-SSF-R-006, Issue 1.2, http://deploy-eprints.ecs.soton.ac.uk/309/, June 2011.

[RD28] Alexei Iliasov, Linas Laibinis, Elena Troubitsyna and Alexander Romanovsky. *Formal Derivation of a Distributed Program in Event B.* In Proc. of ICFEM 2011 -- 13th International Conference on Formal Engineering Methods, October 2011. To appear.

[RD29] Ilya Lopatkin, Yuliya Prokhorova, Elena Troubitsyna, Alexei Iliasov, and Alexander Romanovsky. *Patterns for Representing FMEA in Formal Specification of Control Systems.* TUCS Technical report, http://tucs.fi/research/publication-view/?pub_id=tLoPrTrIlRo11a, March 2011.

2 Summary of Earlier Reported Work

2.1 Knowledge Transfer

The deliverable [RD3] covers the DEPLOY WP3 activities since February 2008 until January 2009:

- Event-B training,
- mini-pilot case studies, and
- preliminary Event-B modelling of BepiColombo (see [RD2]) SIXS/MIXS OBSW requirements.

2.2 Pilot Deployment

The deliverable [RD5] covers the DEPLOY WP3 activities since February 2009 until January 2010:

- continuation of Event-B modelling of BepiColombo SIXS/MIXS OBSW requirements,
- specification of a realistic AOCS (Attitude and Orbit Control System),
- precise Event-B modelling of the AOCS, and
- a lot of Event-B proof activities in association with the modelling activities.

2.3 Measurements and Assessments

With explicit attention to DEPLOY WP3, the deliverables [RD4] and [RD6] cover DEPLOY's official internal measurement and assessment activities since February 2008 until September 2010.

3 Progress of the Enhanced Deployment

In order to respect the spirit of [AD4], it is hereby assumed that enhanced deployment in WP3 consists of the tasks T3.4, T3.5, T3.6 and T3.7 that in [AD4] are described as follows.

3.1 Task T3.4 Addressing sector-specific deployment issues (M24-M48)

3.1.1 Task definition

The greatest challenges in space projects are tracing requirements to the different levels of specifications, integrating requirements resulting from RAMS activities, validating that requirements have been implemented, and making sure that resilience and safety aspects have been properly taken into account. This task can be facilitated by defining a number of generic modelling patterns reusable in different developments in the space domain.

3.1.2 Progress status

In the pilot deployment phase, SSF traced BepiColombo SIXS/MIXS OBSW requirements to SSF's Event-B models of those requirements. In the enhanced deployment phase, no specific attention has been paid to tracing of requirements.

The work described in Section 4.3 is at least closely related to integration of requirements resulting from RAMS activities.

Validation of implementation of requirements has been addressed by means of Event-B machine refinement. WP3 has also made recommendations concerning DEPLOY's code generation activities.

The specification work for the system described in Section 0 has been dominated by resilience and safety aspects.

The modularisation and decomposition plug-ins [RD7] (developed by WP3 members and with WP3 needs in mind) extend the Event-B language with reusable generic modelling patterns.

3.2 Task T3.5 Enhanced deployment (M24-M48).

3.2.1 Task definition

Based on the assessment, the TM/TC software will be subjected to a more enhanced deployment of the formal engineering methods. The scope of interest will be extended to RAMS activities, as they are of paramount importance in critical

systems. Since a large project like the satellite on-board software project will have small blocks that are replicated in different parts of the architecture, feasibility of reuse will be investigated too.
Initial involvement: SSF, Aabo, Newcastle, Southampton

The following two tasks aim at assessing the results of deployment of formal approaches in the space sector and identifying the issues which have to be resolved to ensure success of deployment. Since the development process in the space sector should comply with the sector-specific standards, we should also ensure a smooth integration of formal engineering methods into the existing development process.

3.2.2 Progress status

"TM/TC software" in the text of T3.5 means the same as "Telemetry / Telecommand software (TM/TC software) used in BepiColombo" on the [AD2] page 24. Due to the BepiColombo SIXS/MIXS OBSW work mentioned in Sections 4.2, 4.4 and 0, it is fair to conjecture that in the enhanced deployment phase, the TM/TC software has been subjected to deeper deployment of formal engineering methods than in the pilot deployment phase.

The work described in Section 6 is concentrated on RAMS activities.

Various things have been reused or done with reusability in mind. Feasibility of reuse in a technical Event-B specific sense has been investigated e.g. by experimenting with modularisation and decomposition plug-ins [RD7].

3.3 Task T3.6 Evidence gathering and documentation (M6-M48) and Task T3.7 Integrating formal engineering into existing development process (M24-M48).

3.3.1 Task definition

Task3.6 aims at identifying, collecting and documenting SSF specific evidences about the adoption process of rigorous engineering methods at SSF, given its particular context and domain. It will be performed by the DEPLOY partner within its organization, based on an evidence gathering strategy agreed with the evidence WP (WP11). It will also be supported by this WP in order to yield valuable information for the DEPLOY partner but also for a wider audience outside the project boundaries. More precisely, in the case of SSF, the evidence gathering process will focus on the business model for adoption. In order to be effective a number of enhancements have been identified and must make their way into the modeling method and tools. Evidence of this progress to a point where it can result into a value added service will be gathered. This will cover assessment of a number of "no-go"s (such as language issues, prover power, code generation) and the validation on a second time experiment, potentially carried out on a real request and in parallel with the current way of working. *Initial involvement: SSF, CETIC, Aabo*

Task 3.7 aims at creating guidelines for integrating formal engineering into the development process adopted in the space sector. This work will summarize our experience gained in the deployment.
Initial involvement: SSF, Aabo, Newcastle

3.3.2 Progress status

DEPLOY is so concentrated on Event-B that "rigorous engineering methods" in the text of T3.6 almost certainly refers to Event-B methodology. Let us then use the term "formal method" in the conventional meaning the term is used in computer science publications.

SSF is seriously considering use of formal methods, but there is no really good reason why SSF should choose Event-B for that purpose.

Some people at SSF have done research on formal methods before starting at SSF and can be expected to have a realistic view about use of formal methods. Some people might even like to use formal methods in SSF's projects. There is still no formal method in SSF's projects other than DEPLOY.

A formal method might end up in a project plan if use of the method could be expected to cause an improvement in customer satisfaction or a reduction in costs. A manager may still see formal methods only as risks until there is evidence that companies competing with SSF are beneficially using formal methods.

4 Enhanced Deployment Activities

4.1 DSAOCSS

4.1.1 Background

Control over rotations about vertical, lateral and longitudinal axes is an essential part of any spacecraft or aircraft. As documented and demonstrated by [RD5], [RD12], [RD15], [RD16], [RD17], [RD18], [RD19] and [RD25], DEPLOY WP3 has done a lot of Event-B work on centralised attitude and orbit control systems. Recently this work has been extended to consider decentralised variants of such systems. The work at SSF for that purpose is concentrated on DSAOCSS (Distributed

System for Attitude and Orbit Control for a Single Spacecraft), the requirements of which are presented in [RD27]. The considerations below pay attention to only a few of these requirements and should not be assumed to be exhaustive w.r.t. any single requirement.

4.1.2 Managers

DSAOCSS resembles a distributed PLC (Programmable Logic Controller) system, every manager in DSAOCSS being a local controller that is executed periodically and has specific variables for input from other managers and for output to other managers in such a way that input variables of a manager are never updated in the middle of an execution of the manager. The managers are as follows:

- One AOC (Attitude and Orbit Control) manager that makes all actual AOC decisions, so the actual AOC in DSAOCSS is centralised indeed.
- Five sensor/measurement unit managers that provide data to the AOC manager.
- Two actuator unit managers that are commanded by the AOC manager.

4.1.3 Software beyond DSAOCSS

The managers of DSAOCSS are executed by Data Handling Software (DHSW) that is not a part of DSAOCSS. DHSW also takes care of all actual inter-processor communication. Processor failures and inter-processor communication failures are handled by DHSW, too. This includes a challenging rebooting policy: all processors are rebooted whenever one of them is rebooted.

There is no sporadic inter-processor communication. Instead, every processor is periodically sending data to every other processor. Every message is for updating of certain variables, the set of these variables being fixed for each sender-receiver direction. A receiver can hold at most one message per sender at a time. Overwriting may occur as the hardware of the receiver does not wait for the software of the receiver. Corrupted messages are assumed to exist without ever being passed to the software of DSAOCSS (CRC or other methods are assumed to identify corrupted messages). So, message corruptions in a sense get transformed into message losses.

4.1.4 Unit Management

Every unit is a pair of identical devices, at most one of which is on at a time. The devices are called branches, one being nominal and the other being redundant. A manager of a unit operates on the nominal branch as long as possible. After that, the manager operates on the redundant branch as long as possible. After that, the unit is kept off until rebooting of the whole system.

4.1.5 Mode Management

DSAOCSS has different modes for different needs. A change in the mode typically affects states of the units and usage of AOC algorithms. There is no actual global

mode, but a good approximation of a global mode is achieved by means of a mode synchronization protocol. There is no separate mode manager. Instead, the AOC manager and all unit managers are necessary partners in the protocol. The above-mentioned periodic messaging contains all information needed by the protocol.

4.1.6 How to Model DSAOCSS in Event-B

At the time of writing this, modelling of DSAOCSS in Event-B is a future activity with the following preliminary guidelines:

- Things under responsibility of DHSW (not a part of DSAOCSS) should be kept abstract and simple, still neither too over approximative nor too under approximative.
- The mode synchronization protocol should be modeled early so that considerations on it do not get confused by many other things.
- Event-B without plug-in extensions is the modeling language by default.

4.2 Work on Real-Time

A large number of dependable embedded systems have stringent real-time requirements imposed on them. An analysis of real-time behaviour is usually conducted at the implementation level. However, it is desirable to obtain an evaluation of real-time properties early at the development cycle, i.e., at the modelling stage. Using data processing of BepiColombo SIXS/MIXS OBSW as a working example, WP3 has studied [RD13] possibilities to augment Event-B modelling with verification of real-time properties in the model checking tool Uppaal. In the approach, a process-based view is extracted from an Event-B model and translated into a timed automaton readable by Uppaal. The research has so far managed to demonstrate that the approach makes sense at least in the used example.

Essentially, the developed approach consists of three main stages: (1) "traditional" refinement in Event B; (2) defining an explicit concurrency model called a Process View (PV); (3) and finally, creating a timed-automata model and verification of the desired real-time constraints. Each part of this modelling chain has a specific purpose. The Event-B modelling automated by the Rodin platform facilitates reasoning about functional correctness of the system under construction. A PV model explicitly defines processes and their synchronisation. It is a projection of an Event-B specification that allows us to represent the targeted system architecture and the corresponding communication infrastructure. Finally, a timed automata system model enables verification of the desired real-time properties.

[RD13] presents the formal basis for creating such a modelling chain and discusses the issues associated with its industrial use. It also defines the additional proof obligations needed to formally verify that a PV is a valid projection of an Event-B specification. Moreover, [RD13] demonstrates how to augment a PV model with clocks and timing constraints to arrive at a timed automata model. Indeed, the proof obligations verifying consistency between PV and Event-B models are amenable to the verification by any first order theorem prover, including the Rodin platform. Meanwhile, verification of liveness and desired real-time system properties relies on the widely-used model checker Uppaal.

Note that promotion of real-time tools is not a high-priority goal in this research. Untimed model checking e.g. with ProB [RD7] can well be a reasonable approach in cases where untimed approximation of real-time properties is a reasonable practice.

4.3 Supporting RAMS through FMEA

Operational modes typically have a dominating role in the behaviour of a complex system. Mode consistency and fault tolerance usually have a dominating role in specification of pre- and postconditions for transitions between operational modes. Using Event-B as the formalism and the verbal specification [RD25] of a centralised AOCS (Attitude and Orbit Control System) as a working example, Åbo Akademi University has studied [RD20] possibilities to derive such conditions from FMEA (Failure Modes and Effect Analysis) and to preserve mode consistency in machine refinements. A preliminary methodology has been defined that, to a large extent independently from the application domain, can be applied to control systems in various application domains.

Ensuring dependability and, in particular, fault tolerance of complex control systems is a challenging engineering task. To cope with the complexity, control systems are often developed in a layered fashion, which provides the designers with a convenient mechanism for structuring the system behaviour according to the identified architectural layers. Moreover, the behaviour of a complex system is frequently described and reasoned about using the notion of operational modes. Operational modes can be understood as mutually exclusive sets of the system behaviour. While designing layered mode-rich systems, the developers should ensure mode consistency and guarantee that the mode logic caters to fault tolerance.

In [RD20], it is proposed to conduct Failure Modes and Effect Analysis (FMEA) of each particular mode to identify mode transitions required to implement fault tolerance. The resulting FMEA tables relate a particular mode with different failures (failure modes), the ways to detect them as well as necessary remedial actions. The latter ones are described in terms of new target modes the system should rollback to attempt its recovery. Recovery may also involve dynamic reconfiguration of the system by switching, if possible, some of its failed hardware components to spare ones. Each remedial action is associated with necessary conditions on the states and detected errors of the monitored components of the lower layer.

Moreover, [RD20] demonstrates that, by encapsulating the details of dynamic reconfiguration performed in response to the occurred errors, we can arrive at an efficient mechanism for structuring fault tolerance according to the identified architectural layers.

4.4 Modularisation

4.4.1 Modules and interfaces

Rodin Platform's modularisation plug-in, described and demonstrated in [RD7], [RD10], [RD11] and [RD14], provides facilities for structuring Event-B developments into logical units of modelling, called modules. The module concept is very close to the notion of classical B imports. However, unlike a conventional development, a module comes with an interface. An interface defines the conditions on how a module may be incorporated into another development (that is, another module). The plug-in follows an approach where an interface is characterised by a list of operations specifying the services provided by the module. An integration of a module into a main development is accomplished by referring operations from Event-B machine actions using an intuitive procedure call notation.

4.4.2 General Motivation

Some potential reasons to split a development into modules:

- Structuring large specifications: it is difficult to read and edit large model; there is also a limit to the size of model that the Rodin Platform may handle comfortably and thus decomposition is an absolute necessity for large scale developments.
- Decomposing proof effort: splitting helps to split verification effort. It also helps to reuse proofs: it is not unusual to return back in refinement chain and partially redo abstract models. Normally, this would invalidate most proofs in the dependent components. Model structuring helps to localise the effect of such changes.
- Team development: large models may only be developed by a (often distributed) developer team.
- Model reuse: modules may be exchanged and reused in different projects. The notion of interface makes it easier to integrate a module in a new context.

4.4.3 Experience at SSF

A modularisation plug-in experiment on BepiColombo SIXS/MIXS OBSW requirements was carried out at SSF in August – September 2010 and was focused on a few of the requirements that had been considered in the non-modular experiments reported by [RD5].

The original goals of the experiment were as follows:

- Systematic isolation of activity details and related conditions to modules in such a way that the machines using the modules do not replicate much of what is expressed inside the modules.

- Precision of descriptions of the considered behaviour about as accurate as in the most detailed available non-modular Event-B model.
- Avoidance of massive atomic activities. Long chains of atomic activities do realistically model concurrency.
- To deal with "module integration invariants". Such an invariant refers to variables of more than one module.
- Reasonable total proof effort (including time spent in "iterative optimisation") without compromising the above goals.

The final Event-B project [RD24] of the experiment can be understood to sufficiently meet all the above-mentioned goals, except possibly the proof effort reasonability goal. However, the proof effort was to a certain extent more reasonable than in some earlier Rodin Platform experiments.

Much time in the experiment got spent in recognition and circumvention of bugs and dealing with other undesirable features. The problems that were reported during the experiment (see [RD22] and [RD23]) have been solved in later releases of the plug-in. Since October 2010, the plug-in has been in a good shape w.r.t. the features used in the experiment. In particular, usage of the platform in the modular experiment was not dominated by memory-consumption-based non-response phenomena.

By following certain modelling conventions it is possible to significantly improve the usability of modularisation plug-in.

The conventions are summarized below:

- Avoiding very large and complicated operation post conditions, especially involving existential quantifiers to simplify proofs. In general, complex post conditions can be simplified by introducing additional module variables and invariant properties on these variables

- Refraining from using operation calls to model returned exceptions. To achieve the same effect one might strengthen preconditions of calling events by checking external module variables. Rather than using returned composite values, which can include the status indicating success or a particular occurred exception, additional external module variables storing such a status of the latest call can be introduced

- Avoid generating new values supplied by the environment inside of operation post conditions. The problem can be circumvented by introducing additional local variables in a calling event and then forwarding the value of these local variables as extra parameters of an operation call. Alternatively, module processes can be used for modelling such input from the environment.

4.5 Decompositional Approaches

4.5.1 Atomicity Decomposition

In atomicity decomposition [RD9], coarse-grained atomicity is refined to more fine-grained atomicity. New events introduced in a refinement step are viewed as hidden events not visible to the environment of a system and are thus outside the control of the environment. Any number of executions of an internal action may occur in between each execution of a visible action.

4.5.2 Model Decomposition

The notion of model decomposition [RD9] covers machine and context decomposition. The entry point for the decomposition of a model is a machine and its whole hierarchy of seen contexts. The resulting sub-models are independent of the each other, and the partition of the models includes the allocation of variables, invariants, events and even context elements like sets, constants and axioms to subparts.

Model decomposition has two main styles. In shared variable style, events are partitioned into sub-components in different machines, variables associated with the original events get shared by the machines, and the sub-components can be refined independently but only in such a way that shared variables are present and not data-refined. In shared event style, variables are partitioned into sub-components in different machines, events associated with the variables get split accordingly, and the sub-components can be refined independently without constraints.

4.5.3 Experience at University of Southampton

University of Southampton maintains a decomposition plug-in [RD7] and, as documented and demonstrated in [RD8] and [RD9], has used the plug-in for applying the above-described decompositional approaches to BepiColombo SIXS/MIXS OBSW. SSF's non-decompositional Event-B projects have been used as de facto primary specifications in this experiment that has turned out useful in development of decompositional practices.

SSF has no relevant experience about the decomposition plug-in but sees the above-described approaches being motivated by a several decades long history of similar approaches in formalisms other than Event-B.

4.6 SSF's Internal Event-B Training

4.6.1 Abstract

In January – June 2010, two engineers working for SSF trained themselves to use Event-B. (They got some assistance from experienced Event-B users, but please note that the role of teachers in training at SSF is intentionally marginal in all projects.) The purpose was to find out how much effort is needed for an engineer without background in formal methods to become a competent user of Event-B, and to evaluate the currently available training material, documentation and tools. Those two engineers have later written a report [RD21] about the learning experience. Some of the things said in that report are rephrased below.

4.6.2 Used Tools, Material and Assistance

Rodin Platform with plug-ins [RD7] for text editing (Camille), modularisation and records was used as the Event-B work environment. The training material mostly consisted of the Event-B Wiki, teaching material from the training course held in Zürich in April 2008, and general educational material about formal logic. Assistance was obtained from WP3 team members at SSF, Åbo Akademi University and University of Newcastle.

4.6.3 How Much was Learnt with Which Effort

It has taken the hours of about one working month to attain the ability of working with Event B and Rodin in a reasonably independent fashion. More time would be needed for getting sufficiently familiar with the studied modularisation and records.

4.6.4 Assessment of Material and Tools

The trained engineers were to a large extent satisfied with the used documentation but wished more detailed, searchable and up-to-date tool information and that the Event-B reference material would cover the whole syntax, including proof obligation notations, even on the level of examples.

The basic features of the Rodin environment were found easy to use. For ergonomic reasons and despite of observed usage problems, Camille was preferred to Rodin's default model editor. As usual in any sufficiently long Rodin usage experience, non-response phenomena and other questionable behaviour were observed.

5 Evidence Collection

During the enhanced deployment news contributions were identified to enrich the evidence repository and the associate industrial FAQ. These were jointly discussed with CETIC and SSF. We give here a summary of the main contributions.

5.1 CIF-QAP-1

5.1.1 Question

Can QA practitioner use only a familiar notation (and associated tools) without looking at formal semantics?

5.1.2 Elements of answer

Yes. People use verbal specifications, no matter how simplistic, and learn from concrete examples. In addition domain specific notations can also be used, especially FMEA. With this respect during the enhanced deployment, an integration *of* FMEA into the development flow could take place (see section 6). On-going work [RD29] shows progress on the extraction a formal specification of error detection and recovery procedures from FMEA.

5.2 EM-EA-1

5.2.1 Question

Is it possible to take advantage of formal models beyond using them to guarantee certain properties of a system, for example, to automate or simplify certain development and QA tasks? (code generation, test generation, and deliverable/report/documentation generation are other examples of such exploitation.)

5.2.2 Elements of answer

Yes and no. The activities suggested in the clarification of the question can take advantage of formal models but are still related to creation of confidence and therefore are not really beyond guaranteeing certain properties. In the spirit of exploiting models, SSF is for example considering the mapping to times automata, however it requires to carefully consider the semantic mapping between formalisms.

5.3 EM-EA-3

5.3.1 Question

Is there evidence of formal methods being used to improve the quality of requirements and design documents expressed in less formal notations such as UML?

5.3.2 Elements of answer

Yes and no. Work towards Event-B models has resulted in findings that have been used for improvement of specifications. However, all the findings have been due to inspection and analysis that could well have been done without any formal modelling. The benefit of the formalisation is the ability to drive that inspection in a more systematic way than in a more "traditional" inspection.

5.4 QI-PQAM-1

5.4.1 Question

What impact does the use of formal engineering methods have on the identification of issues at each phase of development cycle? (Here, there are two dimensions to consider: the origin of the defect, and the development stage at which it is detected.)

5.4.2 Elements of answer

Recalling the answer to EM-EA-3, it is perhaps best to consider potential impacts. Then it is clear that usage of formal methods (especially model checking that has not been used much in the SSF deployment) improves detectability of design errors. The phase does not affect the detectability, though ideally all design errors should be caught and eliminated during design phases.

5.5 R-EA-2

5.5.1 Question

Does a selected formal method allow for defining general proof strategies, which can then be reused and specialised to different contexts?

5.5.2 Elements of answer

Yes, though Rodin Platform does not well support automation of more than certain trivial proof reuse strategies.

5.6 TSP-HM-1

5.6.1 Question

What is the cost or effort needed to train engineers/analysts to use a new formalism, taking into account their previous experience with formal engineering methods?

5.6.2 Elements of answer

The report [RD21] written by the two engineers mentioned in section 4.6 is a valuable piece of evidence that gives objective numbers to answer this question.

6 Recommendations and Way Forward

6.1 Way forward

For the remaining time of Deploy SSF will focus on completing the work along the following main directions

Finding a suitable combination of decomposition techniques to achieve an adequate modelling of distributed systems. This work aims at finding viable and scalable solutions for formal modelling and verification of complex distributed systems. The work builds on the latest advances in modelling distributed systems [RD28] and requirements document [RD27]. In [RD28] we relied on a combination of modularisation extension and shared variables decomposition to formally derive and verify a well-known distributed protocol. We believe that modelling of the distributed AOCS [RD28] can be approach in a similar way. It will also add the new challenges arising from handling more complex data structures and dynamically reconfigurable components.

Further work on integrating FMEA into the development flow. The work reported in [RD29] demonstrates how to extract a formal specification of error detection and recovery procedures from FMEA. It also proposes a plug-in that automates application of model transformations defining fault tolerance steps required to cope with failure modes of components. We are planning to investigate how to derive the similar model transformations for the space domain and customize the plug-in to deal with domain specific FMEA. This work will facilitate traceability of requirements from system safety analysis to its formal specification.

Formalization of link between Event-B models and timed automata. In [RD13] we described an experiment in verifying real-time properties of systems modelled in Event-B. The experiment has demonstrated that a tool chain supporting multiple views can be easily constructed from the existing open-source tools. To ensure its safe and justified use in SSF development process we are planning to formally define the link between models representing different views.

6.2 Recommendations

We believe that the success of deployment of formal engineering techniques into the development practice in the space domain could be leveraged by strengthening a link with a "traditional" software development practice. Further research and automation are needed to facilitate a smooth transition from architectural modelling to formal modelling and verification. Moreover, formal verification should provide a prompt and comprehensive feedback at early development stages to facilitate the design space exploration and assessment of design alternatives. Obviously, a mature support for code generation is important for integrating formal methods into the development life cycle. Finally, domain-specific modelling guidelines and tool support automating a significant part of refinement chain are needed to speed up understanding of refinement approach and facilitate its quick

adoption.

7 Conclusions

The work in WP3 has bootstrapped development of both methods and tools of Deploy. The strong background in formal methods has allowed SSF to undertake modelling of systems that are an order of magnitude larger than typical academic examples. This has uncovered several methodological and tooling issues unforeseen at the beginning of Deploy. Firstly, it has shown that proof-based verification becomes feasible only if the models are well-structured, e.g., in a layered manner. Secondly, it demonstrated that neither shared variables nor shared event style decomposition alone can cater to modelling industrial size systems. This has driven research on modularisation extension of Event-B and finding suitable combination of decomposition techniques to create a scalable solution for modelling distributed systems. Moreover, experiments with verification of real-time properties have shown that multiple view modelling and reliance on the existing tools offers a promising solution while integrating modelling of non-functional properties into the refinement chain.