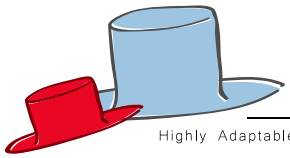

Project N°: **FP7-231620**

Project Acronym: **HATS**

Project Title: **Highly Adaptable and Trustworthy Software using Formal Methods**

Instrument: **Integrated Project**

Scheme: **Information & Communication Technologies**

Future and Emerging Technologies

Deliverable D6.2b
HATS Technology Usage Plan

Due date of deliverable: (T0+42)

Actual submission date: 01-September 2012

Start date of the project: **1st March 2009** Duration: **48 months**

Organisation name of lead contractor for this deliverable: **FRG**

Final version

Integrated Project supported by the 7th Framework Programme of the EC		
Dissemination level		
PU	Public	✓
PP	Restricted to other programme participants (including Commission Services)	
RE	Restricted to a group specified by the consortium (including Commission Services)	
CO	Confidential, only for members of the consortium (including Commission Services)	

Executive Summary:
HATS Technology Usage Plan

This document summarises deliverable D6.2b of project FP7-231620 (HATS), an Integrated Project supported by the 7th Framework Programme of the EC within the FET (Future and Emerging Technologies) scheme. Full information on this project, including the contents of this deliverable, is available online at `http://www.hats-project.eu`.

Deliverable D6.2b presents the results of Task 6.2 (Exploitable Strategy), which is concerned with the preparation of a technology usage plan for the HATS project. The goal of this task was to maximise the opportunities for market adoption of the project results.

The current deliverable builds upon the results of the intermediate deliverable D6.2a, which was published after the first half of the project (T24). It provides all kinds of updates on the market analysis as well as on the description of exploitable items. For example, new exploitable items have been included and planned steps towards exploitation have been provided for all exploitable items. In addition, the deliverable presents the feedback that was obtained from the demonstrations of the HATS ABS framework in several companies.

List of Authors

Karina Villela (FRG)
Sonja Trapp (FRG)
Balthasar Weitzel (FRG)
Taslim Arif (FRG)

The exploitable items and SWOT analyses were created by numerous project participants.

Contents

List of Tables

Chapter 1

Introduction

The aim of this deliverable is to present the technology usage plan for the HATS project. To this end, this chapter provides an overview of the project, discusses the deliverable goals in more detail, and summarises its contents.

1.1 Overview of the HATS Project

As stated in the HATS project proposal, software systems are central for the infrastructure of modern societies and, in order to justify large investments, such systems need to live for decades. This requires software which is highly adaptable. Software systems must support a high degree of (spatial) variability to accommodate a range of requirements and operating conditions, and temporal evolvability to allow these parameters to change over time. Current approaches to reusability and maintenance are inadequate to cope with the dynamics and longevity of future software applications and infrastructures, e.g., for e-commerce, e-health, and e-government. At the same time, we rely increasingly on systems that should provide a high degree of trustworthiness. Thus, the major challenge facing software construction in the next decades is high adaptability combined with trustworthiness. A severe limitation of current development practices is the missing rigor of models and property specifications. Without a formal semantics for developing distributed, component-based systems, it is impossible to achieve automation for consistency checking, enforcement of security, generation of trustworthy code, etc. Furthermore, it does not suffice to simply extend current formal approaches.

The HATS project approach consists of taking an empirically successful, yet informal software development paradigm that supports variability, namely Software Product Line (SPL) Engineering, and put it on a formal basis. The technical core of the project is the Abstract Behavioral Specification (ABS) language that will make it possible to precisely describe SPL features and components, and their instances. Such a precise description will allow analysis of various system properties related to trustworthiness, for instance, security or performance [8]. It will also enable new opportunities for code generation, automatic product derivation, among others. The main project outcome is a methodological and a tool framework achieving not merely far-reaching automation in maintaining dynamically evolving software, but an unprecedented level of trust, once informal processes are replaced with rigorous analyses based on formal semantics.

Therefore, the HATS project will give rise to a fundamental change in the way adaptable, long-lived software is documented, developed, and maintained. It will result in a leap in the degree of automation when systems are faced with changing requirements and it will make it possible to give security and performance guarantees that are at the moment only available for small, closed systems.

Numerous business opportunities will be opened up both for companies that produce and market new tools, as well as for those who use them. In particular, SMEs with highly skilled personnel will be able to profit from a rigorous development methodology.

1.2 Purpose and Structure of this Deliverable

Work Package 6 (*Dissemination and Training*) has the goal of spreading knowledge about the HATS ABS framework in the software engineering community, promoting a long-term perspective on software systems, and emphasizing that investment into more formal and sound approaches will pay off in the long run (Task 6.1 "Dissemination Activities"). This work package also aims to identify the challenges and opportunities of integrating the ABS framework into today's industrial practice (Task 6.2 "Exploitation Strategy"), and to develop and perform training courses and tutorials to educate practitioners in all aspects of the HATS ABS framework (Task 6.3 "Training Program").

This deliverable (D6.2b) presents the complete results of Task 6.2 "Exploitation Strategy". The initial planned deliverable (D6.2) was split into two parts: D6.2a, which was published at an intermediate stage of the project (T24) with the purpose of maximizing the opportunities for market adoption of the project results; and this deliverable (D6.2b), which has the purpose of updating the challenges and opportunities of adopting the project results in industrial settings, by taking into consideration the current set of exploitable items and their current status of development.

The remainder of this document is organized as follows: Chapter 2 provides a market analysis that includes market characteristics, needs and trends; Chapter 3 discusses some related projects; in Chapter 4, all exploitable items of the HATS project are described and analysed according to strengths, weaknesses, opportunities, and threats; Chapter 5 briefly presents the feedback obtained in demonstrations of the HATS approach in companies and events; finally, Chapter 6 summarizes the challenges and opportunities of transferring the HATS ABS framework to the industry.

1.3 Updates in Comparison to D6.2a

In Chapter 2 *Market Analysis*, the list and respective description of SPL Solution Providers have been updated to include the Irish Software Engineering Research Centre (LERO) in Ireland, and the University of Namur in Belgium. Some new references have been included in Section 2.2. Section 2.3 *Current Achievements and Trends* has been completely revised. In Chapter 3 *Related Projects*, a new related project has been included (MBAT Project). In Chapter 4 *Exploitable Items*, the exploitable items have been reordered and the information about the planned steps towards exploitation has been included for all exploitable items. In addition, the exploitable items 4.5 *Provably Correct Compiler*, 4.8 *Delta Modelling Development Workflow*, 4.18 *Learning-based Test Tool*, 4.20 *ABS Product Configurator*, 4.24 *Timed Resource Consumption Analysis and Simulation*, and 4.27 *Product Adaptation Framework* have been included; and the exploitable items 4.8 *Test Case Generation Tool for HATS ABS* and 4.20 *Quantitative Analysis of ABS Programs* from D6.2a have been excluded. We have included the feedback on demonstrations of the HATS framework in Section 5 *Demonstrator and Feedback on Demonstrations.*

Chapter 2

Market Analysis

The HATS project integrates formal models into Software Product Line (SPL) Engineering with the aim to support the development and evolution of highly adaptable and trustworthy software. This chapter provides the results of the market analysis performed in the scope of the HATS project, covering the survey of current providers of solutions in the area of SPL Engineering, the survey of available tools, as well as the survey of related market studies. The elicitation of application domains and problems that would benefit (or have benefited) from a formal approach of such a paradigm and the current achievements and trends related to the integration of these two approaches are also part of the market analysis.

2.1 Market Characteristics

The market for the outcomes of the HATS project can be characterized by the current providers and consumers of solutions in the area of SPL Engineering, as well as by the currently available tools. Solution providers are the subject of the next subsection, as they can become future customers, competitors, or partners of the new formal approach and are easier to identify than consumers. The second subsection is dedicated to SPL Engineering tools. The tooling provided by the HATS project should add value when compared to the currently available SPL Engineering tools, even though this tooling may need major improvements in order to be used in the industry. In addition, two market studies (Subsection 2.1.3 *Market Studies*) were found related to SPL Engineering, which could be purchased after further evaluation of the HATS consortium's members.

2.1.1 SPL Solution Providers

Some major providers of services concerning SPL Engineering, classified by the type of organization, are as follows:

- *Universities*: University of Hildesheim, University of Duisburg-Essen, and University of Bremen, all in Germany; University of Namur, in Belgium.
- *Research Institutes*: Irish Software Engineering Research Centre (LERO), in Ireland; European Software Institute (ESI), in Spain; and Software Engineering Institute (SEI), in USA.
- *Tool Providers*: Pure Systems, in Germany; Big Lever, in USA.
- *Consultants*: metadoc and Software.Process.Management, both in Germany; EVOCEAN, in Switzerland.

The list above does not intend to be complete and excludes members of the HATS consortium on purpose.

Universities

The *University of Hildesheim* has a strong participation in the SPL research community. In 2012, for example, they are going to offer a tutorial at the International Software Product Line Conference (SPLC) about Product Line Engineering for Globalization — PLE4G (`http://www.splc2012.net/Tutorials.html`). One of their main researchers, Prof. Dr. Klaus Schmid, is also one of the authors of the book: *Software Product Line in Action: The Best Industrial Practice in Product Line Engineering* [77]. He will also be participating at the panel discussion on SPL in the Cloud Era during the SPLC 2012 event.

The *University of Bremen* has a research group on Software Evolution. Their work is based on the argument that a SPL often arises from the evolution of an existing single system. Therefore, they particularly work on technologies for supporting the consolidation of variants.

The *University of Duisburg-Essen* also has a strong participation in the SPL research community. They have carried out public and industry projects in this area, addressing different aspects such as requirements, architecture, testing, and variability management. One of their main researchers, Prof. Dr. Klaus Pohl, is the first author of the book: *Software Product Line Engineering: Foundations, Principles, and Techniques* [57].

The research centre PReCISE at the *University of Namur* is also working in the field of SPL engineering and evolution, placing the focus on model checking and verification of software product lines. Their researchers are involved in several projects in this area and have provided a number of publications on this subject.

Research Institutes

The *Irish Software Engineering Research Centre (LERO)* has one competency group on SPL, and one on Formal Methods, both led by Goetz Botterweck. They leverage concepts from model-driven engineering to model the evolution of SPL with the goal to provide support for proactive long-term planning on the feature level.

The *European Software Institute (ESI)* offers training and consultancy services related to systematic reuse and software factories, which include: introduction to systematic reuse, assessment of current reuse approaches, assessment of reuse risks and benefits, support for establishing a reuse based factory, and support for establishing a software product line for embedded systems. Two R&D topics related to SPL Engineering are: variability management methods and tools, and product family evaluation frameworks.

SPL is also a research and consultancy topic at the *Software Engineering Institute (SEI)*. They provide services such as: assessment of the organization's ability to succeed with a SPL approach, a cost model for SPLs, and tailored technical assistance. The latter includes support for: scoping and product line analysis; architecture definition and evaluation; asset mining, migration planning, components development, and testing; development of a product line adoption plan, business case, operating concept, acquisition strategy, and training plans; appropriate data collection, metrics, and tracking mechanisms. Two additional research topics are: (a) production planning and product derivation, and (b) connection of SPLs with other software technologies and paradigms, such as open source, aspect-oriented programming, model-driven development, service-oriented architectures, globalisation strategies, systems of systems, agile development, and ultra-large-scale systems.

Tool Providers

In this section, providers of commercial tools are listed.

Pure Systems (`http://www.pure-systems.com/`) provides a tool for variability management with extension for several well-known tools such as BIRT, CaliberRM, ClearQuest, DOORS, and SAP. The company also provides training and consultation services on variability management assessment, migration to SPL, process analysis, architecture reviews, tailoring of SPLs, among others. SPL is presented as an adventure ("When the time comes and you feel adventurous, you may want to migrate from your current development approach to a (software) product line approach.")

Big Lever (`http://www.biglever.com/`) provides an integrated solution that comprises processes, methods and tools. Product Lines are presented as a simple concept and their tools interoperate with several well-known languages (such as Java, C++, Perl, XML), development environments (such as Eclipse and Visual Studio), and tools (such as DOORS, Subversion, ClearCase, Build Forge, Rhapsody, FrameMaker).

Consultancy Companies

With regard to the consultant companies, *EVOCEAN* is the *Big Lever* partner in Europe and offers typical consultancy services (analysis, evaluation, review and coaching); *metadoc* focuses its services on requirements and technical documentation, and provides a feature modeling tool that integrates with DOORS. *Software.Process.Management* is a small consulting company founded by Andreas Birk, who has active participation in SPL events for the industry. He has organized the different editions of the industry-oriented workshop *Product Line in Context: Technologies, Processes, Business and Organization* (`http://2011.pik-konf.de/index.html`).

2.1.2 SPL Engineering Tools

In this section, SPL Engineering tools with their own web page are briefly described, ranging from commercial tools (*Gears* and *pure::variants*) to tools that have just been proposed (*EMF Feature Model*). The presentation order does not intend to reflect the level of maturity of the tools.

Gears SPL Engineering Tool and Lifecycle Framework is a commercial solution from Big Lever (see Tool Providers in subsection 2.1.1), which offers a uniform variation point mechanism to manage feature-based variations in all stages of the engineering lifecycle, and a product configurator to automatically assemble and configure the customer's assets.

pure::variants Professional & Enterprise supports the development and deployment of product lines and software families. It covers analysis, modeling and implementation, as well as the family deployment activities. Further products, *pure::variants for IBM Rational DOORS*, *pure::variants for Enterprise Architect*, *pure::variants for IBM Rational Rhapsody*, *pure::variants for Simulink*, provide an integrated solution to work with well-established tools while enabling systematic variant management.

XFeature (`http://www.pnp-software.com/XFeature/`) is a feature modeling tool that supports the modeling of product families and applications instantiated from them. *XFeature* is provided as a plug-in for the Eclipse platform and allows users to define their own feature meta-model. Initial development was done under ESA contract 18499/04/NL/LvH by P&P Software GmbH and the Automatic Control Laboratory of ETH-Zürich. The first version of the tool prototype was released in summer 2005. The tool is currently being extended by and used at ETH Zürich in the context of the ASSERT project.

TRAVIS is a product of PURVENTIS (`http://www.purventis.de/content/view/1/2/`) that claims to give a good overview of the causes and effects of the customer's product variance, enable variant scenarios to be drawn up and modified in an effective manner, allow the simulation of the effects of variant decisions, and provide cost models that permit a robust evaluation of individual variants and complete variant scenarios.

FeatureIDE (`http://fosd.de/fide/`) is an Eclipse-based IDE that supports all phases of feature-oriented software development: domain analysis, domain implementation, requirements analysis, and software generation. Different SPL implementation techniques are integrated such as feature-oriented programming (FOP), aspect-oriented programming (AOP), delta-oriented programming (DOP), and preprocessors. The IDE was developed in the Computer Science Faculty of the University of Magdeburg.

FaMa-Framework (`http://www.isa.us.es/fama/?FaMa_Framework`) is a framework for automated analyses of feature models that integrates some of the most commonly used logic representations and solvers proposed in the literature (BDD, SAT and CSP solvers are implemented). It has been designed and developed by the University of Sevilla as a Product Line (*FaMa FM Core* and *FaMa Extensions*). *FaMa-Framework* claims to be the first tool integrating different solvers to support automated analyses of feature models.

Moskitt Feature Modeler (`http://www.pros.upv.es/labs/index.php?option=com_content&task=view&id=51&Itemid=35`) is a free open-source tool for editing feature models. It can be used standalone as an Eclipse

plugin or integrated in the *MDE Moskitt Environment*. It has been developed by ProS Labs based on the Eclipse Modeling Framework (EMF), the Graphical Modeling Project (GMF) and ATL, a model transformation technology.

RequiLine (`http://www-lufgi3.informatik.rwth-aachen.de/TOOLS/requiline/`) is a requirement engineering tool whose goal is to support the efficient management of product lines. It enables one to model the product line using features and requirements, and to derive product configurations from the specified model. Additionally it contains a consistency checker, a query interface, user management with different views, and an XML interface. It was developed as part of a PhD thesis of RWTH Aachen University.

SPLOT (`http://www.splot-research.org/`) is a Web-based reasoning and configuration system for SPLs that offers a set of state-of-the-art tools targeting academics and practitioners in the field. The system benefits from mature logic-based reasoning techniques such as SAT solvers and binary decision diagrams (BDD) to provide efficient reasoning and interactive configuration services. It is possible to edit, debug, analyze, configure, share, and download feature models instantly. In addition, SPLOT provides a feature model repository containing real and generated models to encourage knowledge sharing among researchers in the field.

The Christian Doppler Laboratory for Automated Software Engineering (`http://ase.jku.at/modules/product-lines/index.html`) is developing an automated approach for product line engineering in cooperation with Siemens VAI. It is composed of $DOPLER^{VM}$, a flexible and extensible approach for variability modeling, and $DOPLER^{UCon}$, a user-centered approach for product configuration. The supporting tools are: *DecisionKing*, for variability modeling and management; *ProjectKing*, which guides and supports the product derivation and sales process; and *ConfigurationWizard*, which provides capabilities for product customization, requirements elicitation, and configuration generation.

EMF Feature Model (`http://www.eclipse.org/proposals/feature-model/`) is an open source project proposed under the Eclipse Modeling Framework Technology Project (EMFT). The project aims to define a standard representation of feature models inside the Eclipse platform. The intention is to provide a uniform representation for variability information for tools based on the Eclipse Modeling Framework. This will allow easy and consistent access to variability-related information, such as variation points and variant decisions, in DSLs, M2M transformations, and other contexts where variability information is produced or consumed.

2.1.3 Market Studies

Only two market studies were found when trying to answer the questions: How does the software market currently support the development of adaptable but trustworthy software? Are SPL Engineering and formal models included in the current solutions? Both are related to embedded systems and none makes reference to any kind of formal specification.

- The market study *Software development in the product lifecycle* [56] includes SPL Engineering in its table of contents. It costs about 1850 Euros.
- The report *Oki Eletric Industry Co. Ltd.—Financial and Strategic Analysis Review* [54] includes among the recent company's developments: Oki unveils the new MoBiz software product line for smartphones. Oki is a provider of info-telecommunication systems and MoBiz is their mobile solution for companies that use smartphones as business devices. The report price is about 90 Euros.

The HATS consortium has decided not to buy any of these reports because they are not close enough to the scope of the project.

2.2 Promising Application Domains, Relevant Concerns, and Market Needs

In a survey of recent reports on the application of SPL Engineering with formal specification, we found that these technologies have been jointly applied in the domains of:

- insurance sales [37]
- power generation plants [71]
- ambient intelligence [25, 3]
- telecommunications [73, 39]
- avionics [69]
- medical devices [35]
- automotive [53]
- autonomous space exploration missions [29, 30]
- industrial automation or production [21, 27]
- video surveillance [1].

From this list, it is possible to assume that the integration of SPL Engineering and formal models can be positively exploited in any application domain in which software systems share a lot of commonalities (variant-rich software systems) and are business or safety-critical.

According to Becker et al. [7], one of the most prominent domains is the automotive industry. Due to the fact that the driving factors for innovation and unique selling points of today's cars are software systems, handling variants and reuse in a systematic way is indispensable. Consider, for example, a car which the customer can order either as a two-wheel or four-wheel drive variant. One can easily imagine that all the control algorithms for the traction-control need to be designed in a different way for each variant. Furthermore, from a safety point of view, different critical situations (or hazards) may occur when four wheels are driven instead of two wheels. At present, each of these variants must be considered as a separate product and, for each variant, the whole development process, including especially the safety engineering activities and artefacts, must be re-applied mostly from scratch. With the new international standard ISO 26262 [34] for functional-safety of road-vehicles, the compliance with the prescribed safety-engineering process is mandatory for the car manufacturers. Similar challenges are arising in the commercial vehicle domain. Due to the awareness for safety generated by ISO 26262, other automotive domains such as tractors and machinery for agriculture and forestry have also framed an international standard for functional safety, ISO 25119 [33]. This domain is facing the challenge that the maturity of their traditional safety activities is quite low and the maturity level of their process is often lower than that of the automotive industry. Furthermore, the ratio between number of variants and overall units sold is an order of magnitude higher than in the road-vehicle industry.

Becker et al. [7] also point to the domain of medical devices. This domain, with an uncompromising demand for safe products, is also facing the software challenge similar to the automotive industry. More and more functionality is based on software, and different products often lie only a set of parameters apart from each other. Especially challenging in this area is the condition mentioned in the IEC 62304 standard [32] that if a dangerous situation can arise as a result of a software fault, the likelihood that the fault will arise is considered to be 100 percent. Hence, when software is part of a medical device, one cannot reason quantitatively about meeting the safety requirements. New approaches for qualitative reasoning via semantics of elements and not via numbers need to be exploited. In fact, this challenge is not particular for the medical device domain, but for all products where software is part of the overall system.

From another perspective, formal models in SPL Engineering have been commonly used to address problems such as: consistency checking [38, 19, 41, 1], complexity [81, 73, 27, 42], evolution [19, 35, 31], variability in security requirements [46], automated software development [25, 44], and the already mentioned failure detection and safety-critical engineering [69, 35, 7]. In this sense, PL Engineering associated to formal methods can also be exploited when the family of systems is not business or safety-critical, but when one of the aspects above are of high interest.

However, two issues concerning the adoption of formal models by the software engineering industry were found in the recent reports. Kim et al. [38] point out that formal methods are hard to understand by non-experts. Becker et al. [7] argue that semi-formal models (e.g., using UML) have become widely known and are the quasi standard in SPL Engineering as well as in Safety Engineering. They conclude that the integration of these two engineering processes should be based on a semi-formal representation as an intermediate step in the continuous process towards rigorous formal models and methods.

Furthermore, Clarke and Proença [15] emphasize the need for scalability, as software development involves increasingly larger feature models; and modularity, as many stakeholders have a vested interest in different aspects of a feature model and modularity techniques can be used to independently express their views.

Therefore, lightweight formal approaches that can be easily understood and integrated into the current practices, and that are scalable and modular would be welcome by practitioners. In terms of tool support, Chen and Babar [12] claim that practitioners expect to have integrated, standardized, and end-to-end tool support, instead of having different tools for closely related problems, and such requirements have been not met yet.

2.3 Current Achievements and Trends

There are some major IT-trends under way: cloud computing [78], cyber-physical systems [58], and the internet of things [5]. A prerequisite for cloud computing is the ability to abstract away from physical resource allocation, load distribution, the architecture of the execution platform, among other concrete aspects. However, the question is: How does one specify intended behavior without referring to concrete resources? Likewise, the emergence of cyber-physical systems and the internet of things emphasize the need for abstract behavioral description of highly configurable and diverse systems. The HATS project has the opportunity to address the challenges posed by these major trends and, consequently, benefit from them.

In addition to the major trends in the IT market, we have considered achievements as early signs of potential trends and we have tried to draw conclusions on potential trends from the analysis of papers published at the last three International Software Product Line Conferences (SPLC), the main forum of discussion on SPL Engineering.

2.3.1 Achievements as Sign of Potential Trends

An article on "Formal Methods in Software Product Line Engineering" written by members of the HATS consortium was published in IEEE Computer in 2011 [64]. This is a sign of the increasing interest of a wider community in the approach of the HATS project. The authors argue that SPL Engineering should be cast as a model-centric development process relying on a uniform formal modeling framework.

The second relevant achievement is the series of workshops on Formal Methods in SPL Engineering (FMSPLE), which will have its third edition in 2012. The first edition took place in 2010 and was initiated and organized by members of the HATS project. In 2011, FMSPLE and a workshop on Automated SPL Engineering (ASPL)[1] were merged in order to increase the awareness of the community towards the integration of formal approaches into SPL engineering. All editions of the FMSPLE workshop have been held in conjunction with SPLC.

2.3.2 SPLC Papers as Indicators of Challenges and Opportunities

In order to find some indicators of trends in the scope of the HATS project, we have analyzed the papers published at SPLC related to the usage of formal models in SPL Engineering and to SPL for Information Systems. Our analysis covered the main conference and its workshops from 2010 to 2012[2]. In terms of

[1] ASPL took place at SPLC 2008 having formal methods as a hot topic.

[2] Regarding 2012, only the titles of the papers were available before the due date of this deliverable.

workshops, the three editions of the FMSPLE workshop (2010-2012) and the two editions of the SCArVeS workshop[3] (2011-2012) received special attention. Table 2.1 summarizes our findings.

Table 2.1: SPLC Papers related to HATS Scope

Area	Topic	(M)ain Conference/ (W)orkshop
Product Line Context	Use of VDM++ to describe features [74]	M
	Use of propositional logic to represent the structure and the constraints of a feature model, as well as the representation of soft constraints in fuzzy logic [6]	M
Model Driven Product Line	Use of a consistent, unified formalism for models, code, and configuration in order to improve variability management and traceability [80]	M
	Incremental automata-based approach for software product line model checking [16]	W
	Two lightweight extensions to Petri nets to model and verify SPLs [48](HATS paper)	W
	Use of Debian repositories for the extraction of orthogonal variability models [23]	W
Feature-Oriented Software Development	Raising awareness of the problem of abstract features for different kinds of analyses on feature models [72]	M
	Intersection of feature models [76]	W
	Formal approach for merging feature models [10]	W
	A theory of views for feature models [15](HATS paper)	W
	Optimizing the product derivation process by using techniques to find good feature selection sequences [13]	M
Service Oriented Product Line	Study on service identification methods for software product line [75]	W
	Use of the linear logic for matching of service feature models [49]	W
	A method for designing and implementing context-aware autonomous web services in system families [2]	M
	Use of software product lines for cloud computing applications [60, 11, 66]	W
	Combining service-oriented architectures and software product lines [47, 68, 59, 51]	W
Delta Oriented Product Line	Use of delta-oriented programming of software product lines [62](HATS paper)	M
	Transformational proof system for delta-oriented programming [17](HATS paper)	W
	Use of modal logic for abstract delta modeling [18]	W
	Methodology for SPL Engineering based on formal methods [14](HATS paper)	W
	An industrial case study of the HATS approach with respect to expressiveness, scalability, and usability [82](HATS paper)	M

[3]Workshop on Services, Clouds, and Alternative Design Strategies for Variant-Rich Software, also co-organized by HATS members.

Varibility Management	Variability management in a single logical framework consisting of a Modal Transition System (MTS) and a set of formulae [4]	M
	Formal transformations between feature model and decision model [22]	M
	Use of propositional logic in variability models and binding decisions [61]	W
	Formal syntax and semantics for documenting variability in activity diagrams based on Petri-nets [28]	M
Automated Analysis	Automated analysis of conflicts based on visual model language for contracts and deontic contract language [45]	W
	Automatic derivation of a product performance model from a software product line model [70]	M
	An executable algebra for product lines [26]	W
Quality Assurance	Integrating software safety and product line engineering using formal methods [7](Position paper)	W
	Methodology to apply combinatorial testing to a feature model, based on the translation of the feature model into a binary Constraint Solving Problem (CSP) and the usage of pairwise testing [55]	M
	A method for automated bug location, based on a Boolean Constraint Propagation (BCP) algorithm for non-clausal formulas [50]	M
	Use of formal methods to gain confidence in the correct behavior of a space exploration mission [30]	M
	Verification of real-time software product lines [43]	M
	Method and tool for detecting errors in feature models [24]	W
Product Configuration and Derivation	A unified perspective for users configuring products in multi product line environments, regardless of the different modeling methods and tool used [20]	M
	Using an incremental algorithm and tool-support for automatically optimizing user guidance [52]	M
	Quality variability aware product configuration [79](HATS paper)	W
	Optimization of product instantiation using Integer Programming [9]	W
	Dynamic configuration management for cloud-based application [67]	W

Taking into consideration all aforementioned contributions, some general conclusions on trends can be drawn:

1. Product configuration and derivation
   - Approaches to consider not only the functional features but also context, soft/hard constraints, abstract features, and runtime environment [74, 72, 6, 2]
   - Approaches to guide the configuration, e.g. by ordering the set of decision-making questions or prioritizing the selection of features [52, 13, 79]

- Early stage prediction of product performance [70, 79]

2. Programming paradigm
   - Delta-oriented programming [62, 18, 17]
3. Product line verification
   - Incremental approach [55, 16]
   - Verification of critical and/or real-time product lines [30, 43]
4. Product lines for cloud computing application and services [2, 66, 75, 11, 67].

In total 44 publications from the software product line community are related to the scope of the HATS project since 2010. Six papers out of these publications were published by members of the HATS consortium. It means above 85 percent of the publications comes from externals, which is a great sign of the interest of the community in SPL Engineering based on formal methods. One third of the papers (15 papers) applied their methods in different application domains.

Chapter 3

Related Projects

Other relevant information when performing a market analysis concerns related projects. On the one hand, the identification of synergies with related projects may shorten the time frame required to obtain the project results, especially if some kind of cooperation can be established. On the other hand, if related projects should be considered as competitors for any reason, the threat that they deliver relevant results before our own project should be taken into consideration. This chapter presents four projects that could be considered as projects related to HATS.

RODIN (*Rigorous Open Development Environment for Complex Systems*) was an EU-IST project (511599) that ran from 2004 to 2007. The objective was the creation of a methodology and supporting open tool platform for the cost effective rigorous development of dependable complex systems and services. The focus was on tackling complexity caused by the environment in which the software is to operate. The unified methodology combines formal methods with fault tolerance techniques, supports structured refinement and decomposition, as well as reuse of existing assets. The project approach was to extend existing formal methods with generic mechanisms to support component reuse and composition. The *RODIN* environment has been used in a number of industrial case studies within and outside the project: air-traffic information display system, a railway interconnect system, an engine failure management system, an ambient information system, and mobile telecom protocols. The *RODIN* project shares with the HATS project the fact of combining formal methods and reuse of existing assets.

CESAR (*Cost-Efficient methods and processes for SAfety Relevant embedded systems*) was an ARTEMIS joint undertaking project (`http://www.cesarproject.eu/`) that ran from 2009 to 2012 to address the need for ultra-reliable embedded systems in the context of the automation domain and three transportation domains (automotive, aerospace, and rail). *CESAR* aimed to boost cost efficiency of embedded systems development, and safety and certification processes by providing:

- a so called Reference Technology Platform (RTP), which is a systems and software level environment for the development, validation, and verification of requirements and architectures for safety-critical hard-real-time system development;
- a model-driven process for the compositional development of safety critical systems based on the RTP, where the analysis of the extra-functional aspects is integrated into the primary functionality, thereby allowing an integrated design of the entire system;
- an analysis process to establish an industrially applicable methodology for exploration of design spaces, multi-criteria constraint satisfaction and design decision making, with particular regard to safety and diagnosability properties.

.

The *CESAR* project is related to the HATS project, because formal models are covered by the RTP and there was a specific Task Force dedicated to SPL Engineering.

The ARTEMIS joint undertaking project MBAT (`https://www.mbat-artemis.eu/home/`) also focusses on the transportation domains. MBAT stands for "Combined Model-based Analysis & Testing of Embedded Systems" and it is going to run from 2011 to 2014. It aims at combining model-based testing technologies with static analysis techniques leading to a new validation and verification method to accompany the embedded systems development process. Similar to the *CESAR* project, a Reference Technology Platform (MBAT RTP) will be used to enable the production of high-quality and safe embedded systems. The Learning-based Test Tool developed in the HATS project (see Chapter 4, Exploitable Item 4.18) will be made available to industrial partners from the MBAT project for evaluation purposes.

SecureChange (`http://www.securechange.eu/`) was a EU-FP7 project (231101) that ran from 2009 2012. It objective was to develop techniques and tools that ensure "lifelong" compliance to security, privacy and dependability requirements for long-running evolving software systems. The project developed processes and tools that support design techniques for evolution, testing, verification, re-configuration and local analysis of evolving software. The focus was on mobile devices and homes. The achievements included:

- architectural blueprint and integrated security process for lifelong adaptable systems
- methodology for evolutionary requirements with tools for incremental requirements models evaluation and transformation
- security modelling notation for adaptive security with formally founded automated security analysis tools
- IT security risk assessment with tool-support for lifelong adaptable systems
- techniques and tools to verify adaptive security while loading on-device
- model-based testing approach for evolution.

Verification techniques and tools, security modelling, and support for the development of lifelong adaptable systems are topics of interest shared by *SecureChange* and HATS. The synergies between the two projects were supported by the EternalS Coordination Action (`https://www.eternals.eu/`). This Coordination Action aims to coordinate research in the area of trustworthy eternal systems and provides a platform for mutual awareness and cross-fertilization among four "ICT Forever Yours" projects: *CONNECT*, HATS, *LivingKnowledge* and *Secure Change*. *CONNECT* is also a project related to HATS. It aims to enable continuous composition of networked systems to respond to the evolution of functionalities provided to and required from the networked environment. The synergies between these two projects have been exploited in three joint papers: "Software diversity: state of the art and perspectives" [65], "Comparing structure-oriented and behaviour-oriented variability modeling for workflows" [40], and "A constraint-based variability modeling framework" [36].

LivingKnowledge, however, cannot be considered as a project that is closely related to HATS. Despite of being based on the vision that diversity is an asset and should be traceable, understandable and exploitable, the goal of *LivingKnowledge* is to improve navigation and search in very large multimodal datasets.

Chapter 4

Exploitable Items

In this chapter, each exploitable item derived from the project results is described and analysed according to strengths, weaknesses, opportunities, and threats.

4.1 Documentation Structure

This first section presents the structure used to describe all exploitable items.

Description: The description allows one to quickly understand the contents of an exploitable item.

Intended Market or Sector: Some anticipated market where the item can be exploited.

Application Domains: Application domains which the exploitable item is specially relevant to. They may constrain the intended market.

Usage Scenarios: Taking the intended market and application domain into consideration, it contains typical usage scenarios that might occur. They are not intended to cover all possible uses of the item.

Final Status: Description of the state that the element will have at the end of the project. Possible values for that status are the following:

- High level concept: Some kind of sketch for using the exploitable item is available, but the details are not clear.
- Fine grained concept: The concept is ready to be applied, but it has no tool support or training material.
- Prototype for proving concepts: A proof of concept is available, but it might contain errors and it is not yet appropraite for scale-up to industrial applications. In this context, a prototype is not only a piece of software, a method with corresponding documentation is also a prototype.
- Prototype for internal usage: The proof of concept was applied in at least one real world example.
- Prototype for external usage: The implementation of the concept is usable for non-project members and it was applied in real world scenarios.
- Product: There is a product available which is ready to be sold, training material is available and packaged.

Planned Steps Towards Exploitation: Steps planned for after the end of the project in order to be able to better exploit the item.

Intellectual Property Protection: If there are legal contracts or patents concerning the future usage of the exploitable item.

Owner and Other Partners: It reflects the distribution of effort. Some project members are the leading developers of the exploitable item. Other partners have been involved without actually leading it.

Main Contact: If further usage or development of the item is intended, the main contact can provide additional information and coordinates future activity.

Related HATS Deliverables: At least one HATS deliverable describes the item in more detail. In most cases there are several deliverables that address the exploitable item.

Table 4.1: Exploitable Items

No.	Name	Application domain	Status	Owner(s)	Deliverables
4.2	ABS Language and Core Tools	No specific domain	prototype for external usage	All HATS consortium members	D1.1a, D1.1b, D1.2
4.3	Location Type System	Any software system modelled with ABS	Prototype for internal usage	UKL	D1.2
4.4	Deductive Compilation	Any software system modelled with ABS	Prototype for proving concepts	TUD	D1.3, D1.4, D2.5
4.5	Provably Correct Compiler	Any software system modelled with ABS	Prototype for proving concepts	IOC	D1.4
4.6	Feature Modelling Framework	No specific domain	Prototype for internal usage	KUL	D1.2
4.7	Delta Modelling Framework	No specific domain	Prototype for external usage	KUL, TUD and others	D2.2
4.8	Delta Modelling Development Workflow	No specific domain	Prototype for internal usage	KUL, TUD and others	D2.2
4.9	Models and constructs for components	No specific domain	Prototype for proving concepts	BOL, UKL	D2.1, D3.1.a, D3.1.b
4.10	Bahavioural specification framework	No specific domain	Fine grained concept	UKL, CWI	D1.2, D2.5, D2.6
4.11	HATS Methodology	Any long-lived software system	Prototype for proving concepts	All HATS Consortium Members	D1.1b, D1.5
4.12	Integrated Development Environment	Any software system modelled with ABS	Prototype for proving concepts	UKL, UIO	D1.1, D1.2, D1.3, D1.4, D1.5
4.13	Formal verification tool	Any software system modelled with ABS	Prototype for internal usage	TUD, UIO	D1.3, D2.5, D4.3
4.14	Interactive simulator	Any software system modelled with ABS	Prototype for proving concepts	UKL	D1.1-D1.5, D2.3
4.15	Partial evaluation-based test case generator	Any software system written in Java or modelled in ABS	Prototype for internal usage	UPM	D2.3
4.16	Monitor inlining tool	No specific domain	Prototype for proving concepts	KTH	D3.4
4.17	Visual debugging tool	Any software system modelled with ABS	Prototype for proving concepts	TUD	D1.3, D2.3
4.18	Learning-based test tool	Reactive and embedded systems	Prototype for external usage	KTH	D2.3

4.19	SPL verification tool	Software for which it is critical how it accesses resources over time	Prototype for proving concepts	KTH	D2.5, D4.3
4.20	ABS product configurator	No specific domain	Prototype for internal usage	FRG, UPM, NR	D4.4
4.21	Static analysis techniques for deadlock freedom	No specific domain	Fine grained concept	BOL	D2.4, D2.5
4.22	Cost and termination analyzer	Any software system written in Java or modelled in ABS	Prototype for internal usage	UPM	D4.2
4.23	Automated derivation and verification of resource bounds	Any software system modelled with ABS	Prototype for proving concepts	UPM	D1.3, D4.2
4.24	Timed Resource Consumption Analysis and Simulation	No specific domain	Prototype for external usage	UIO	D2.1
4.25	Hybrid analysis for evolvability	Domains where trustworthiness must be assured	Fine grained concept	KUL	D3.3
4.26	Model mining algorithm and tools	No specific domain	Prototype for proving concepts	KTH, NR, UPM	D3.2
4.27	Product adaptation framework	No specific domain	Prototype for external usage	FRG	D3.5

4.2 ABS Language and Core Tools

Description

An object-oriented language featuring asynchronous communication and collaborative concurrency control. It is the common framework for modelling systems inside the HATS project, and will be used as basis for the different analysis techniques and tools developed inside the project. The core tools of the language are:

- A standard compiler frontend, including a parser and type checker.
- A Java backend to generate Java code and execute ABS models using the JVM.
- A Maude backend to generate Maude code and simulate ABS models using Maude.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users. The concepts are suitable for usage in education.

Application Domains

There is no specific application domain intended, usage in a broader sense is possible.

Usage Scenarios

Specification and modelling of software systems. The usage of ABS will enable the application of the analysis techniques and tools developed inside the HATS project.

In addition, it will be used at the university in software engineering and formal methods courses at graduate and postgraduate level.

An ideal way of exploiting the ABS methodology is designing lecture notes to teach (a subset of) novel concepts by using the ABS language. A common format would be the one used at a summer school of 10 lessons of about 90 minutes each. To this end a HATS summer school is going to take place in Bertinoro, in the period of 24-28 September 2012.

Final Status

The item will be available as a prototype for external usage.

Planned Steps Towards Exploitation

- Apply ABS tools in larger scale Product Line Engineering context
- Use ABS tools as basis of future project proposals in alternative domains (e.g., cloud computing)
- Improve quality of implementation, in particular, efficiency of generated code.

Intellectual Property Protection

Currently there is no intellectual property protection planned.

Owner and Other Partners

All HATS consortium members were involved in the creation of the ABS Language and its core tools.

Main Contact

Reiner Hähnle {`haehnle@cs.tu-darmstadt.de`} (TUD)

Related HATS Deliverables

D1.1.a, D1.1.b, D1.2

SWOT Analysis

See Table 4.2 (common analysis for exploitable items 4.2, 4.12, 4.14)

Table 4.2: SWOT Analysis *ABS Language and Core Tools*

Table 4.2: SWOT Analysis *ABS Language and Core Tools* (continued)

	Strengths	**Weaknesses**
	Consortium: • Know-how in the development of object-oriented, concurrent, and distributed languages • Know-how in conducting empirical studies *ABS language:* • Formal semantics • Designed to support verification • Similar to existing programming languages that are used in practice • Based on an already existing and used language, which was developed by one of the consortium members • Applicability shown in several case studies	*Consortium:* • Not enough manpower to implement industry-strength tools *ABS language:* • ABS language is not known to the community • Documentation is designed for researchers
Opportunities • There is a demand for adaptable and trustworthy software • There is a demand for model-based software development • To the best of our knowledge there are no comparable languages • There are no approaches in the product line community working with behavioural models that are close to the code level • Use of the ABS language and tools in acquiring further projects	**Strengths × Opportunities** • Demonstrations of the ABS language to companies that are interested in adaptable and trustworthy software, in model-based software development and/or product line engineering • Use of the case studies for demonstration	**Weaknesses × Opportunities** • Disseminate the ABS language in the industry community • Emphasize that the ABS language is similar to existing languages to improve its acceptance

Table 4.2: SWOT Analysis *ABS Language and Core Tools* (continued)

Threats	**Strengths × Threats**	**Weaknesses × Threats**
• The work depends on external funding • The industry does not recognize the benefits of the language compared to existing languages and modelling techniques • The cost-benefit relation must be positive • The industry requires high-quality tools	• Provide benchmarks to show the benefits of the ABS language • Provide empirical evidence on the cost-benefit relation	• Provide documentation for end-users so that companies can adopt the language more easily • Search for partners that can implement industry-strength tools and/or make the software open-source to benefit from external developers

4.3 Location Type System

Description

A type system extension for ABS that statically distinguishes far from near object references.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users. The concepts are suitable for usage in education.

Application Domains

Any software system that is fully or partially modelled with ABS can make use of this item.

Usage Scenarios

The type system is realized as a pluggable type system extension to the standard ABS type checker. ABS modellers use this extension to guarantee that synchronous method calls are not done on far object references.

Another scenario is to use the static knowledge to support the verification of ABS models.

Final Status

The item will be available as a prototype for internal usage.

Planned Steps Towards Exploitation

- Application of the type system to other settings and domains (e.g. Java RMI)
- Improvements in the tool chain
- Application of the tools to larger case studies

Intellectual Property Protection

BSD 3-Clause New or Revised License

Owner and Other Partners

UKL

Main Contact

Arnd Poetzsch-Heffter {poetzsch@cs.uni-kl.de} (UKL)

Related HATS Deliverables

D1.2

SWOT Analysis

See Table 4.3

Table 4.3: SWOT Analysis *Location Type System*

	Strengths	**Weaknesses**
	• Know-how in type systems for object-oriented languages • The type system gives static guarantees	• The scope of the type system is limited to the concurrency model of concurrent object groups
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• There is no existing comparable type system for an object-oriented language with the concurrent object groups model of concurrency	• Show, by means of examples, that the type system is expressive enough to handle realistic models/programs	• Try to generalize the approach to other concurrency models • Convince the research community that the technique is interesting even outside ABS
Threats	**Strengths × Threats**	**Weaknesses × Threats**
• Several type and effect systems exist in the literature	• Clearly work out the differences with existing type systems	• Provide an easy to use implementation

4.4 Deductive Compilation

Description

A compiler from ABS to Java bytecode that realizes strong optimization techniques involving first-order deduction and partial evaluation. It is based on the symbolic execution engine for ABS that is part of KeY ABS (see *Visual Debugging Tool for HATS ABS* in Section 4.17).

Intended Market or Sector

In general all companies that do their business with software engineering are potential users. The concepts are suitable for usage in graduate and postgraduate education.

Application Domains

Any software system that is fully or partially modelled with ABS can make use of this item.

Usage Scenarios

The compiler can be used for automated generation of optimized Java bytecode from ABS models. It is possible to instantiate the variation points in ABS models in order to obtain optimized code for the selected instances. In addition, it will be used at the university in graduate and postgraduate courses concerning advanced compilation techniques.

Final Status

The item will be available as a prototype for proving concepts.

Planned Steps Towards Exploitation

It is planned to develop the prototype further into a compiler framework for generation of correct and optimized code. This will be pursued in follow-up projects.

Intellectual Property Protection

The tool is expected to be released under the GPLv3.

Owner and Other Partners

TUD

Main Contact

Reiner Hähnle {haehnle@cs.tu-darmstadt.de} (TUD)

Related HATS Deliverables

D1.3, D1.4, D2.5

SWOT Analysis

See Table 4.10 (common analysis for exploitable items 4.4, 4.5, 4.13, 4.17, 4.23)

4.5 Provably Correct Compilation

Description

A compiler from ABS to Scala that supports distributed execution. It uses the continuations plugin for Scala to implement COG-internal concurrency and the Akka actor library to implement inter-COGs concurrency. The compilation methodology is verified using the Agda proof assistant (on-going work).

Intended Market or Sector

In general all companies that do their business with software engineering are potential users. The concepts are suitable for usage in graduate and postgraduate education.

Application Domains

Any software system that is fully or partially modelled with ABS can make use of this item.

Usage Scenarios

The compiler can be used for automated generation of Scala code from ABS models which can then be compiled with the standard Scala compiler and executed on the Java virtual machine.

Final Status

The item will be available as a prototype for proving concepts.

Planned Steps Towards Exploitation

It is planned to develop the prototype further into a compiler framework and to scale-up and complete the Agda formalization. This will be pursued in follow-up projects.

Intellectual Property Protection

The tool is expected to be released under GPLv2.

Owner and Other Partners

IoC

Main Contact

Tarmo Uustalu {tarmo@cs.ioc.ee} (IoC)

Related HATS Deliverables

D1.4

SWOT Analysis

See Table 4.10 (common analysis for exploitable items 4.4, 4.5, 4.13, 4.17, 4.23)

4.6 Variability Modelling Framework

Description

The variability modeling framework consists of three main languages: 1) The feature modelling language μTVL, which provides a textual language and associated tools for describing and reasoning about feature models; 2) The product line configuration language (CL), which links feature models specified in μTVL with delta modules (4.7) to provide a specification of the variability in a product line; 3) the product selection language (PSL), which specifies a product by stating which features are to be included in the product and by setting any attributes of those features with concrete values.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users. The concepts are suitable for usage in undergraduate, graduate and postgraduate education.

Application Domains

There is no specific application domain intended, usage in a broader sense is possible.

Usage Scenarios

μTVL is designed to be used in the requirements analysis and design phases for software product line engineering. Software developers can use μTVL to describe their feature models and attributes associated with eventual deployment platforms. The associated tools enable validity of selected configurations to be tested, along with exploration of the design space using constraint satisfaction techniques.

Product line engineers can use CL to describe product lines by specifying the features that are part of the product line and also the relations between those particular features and corresponding deltas.

Application developers can use PSL to specify the features that will be part of the product. PSL can be further used to derive the actual product.

Final Status

The item will be available as a prototype for internal usage.

Planned Steps Towards Exploitation

μTVL, CL and PSL have been incorporated into the ABS tool suite, so it can be exploited by any parties using ABS. There are plans for developing an Eclipse plug-in for μTVL, CL and PSL which could be exploited to enhance the ABS tool suite.

Intellectual Property Protection

μTVL derives from TVL, developed at University of Namur, so potential IP conflicts may arise there.

Owner and Other Partners

KUL (owner), CWI, UIO, FRH, UKL

Main Contact

Dave Clarke {dave.clarke@cs.kuleuven.be} (KUL)

Related HATS Deliverables

D1.2

SWOT Analysis

See Table 4.4

Table 4.4: SWOT Analysis *Feature Modelling Framework*

	Strengths	**Weaknesses**
	• A solid background in theoretical techniques with skill in applying them to practical problems • μTVL, CL and PSL have a formal foundation	• Limited experience with large scale software product line development • Limited contact with companies using software product lines
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• Software product line engineering is increasingly becoming a technique for companies to consider • Feature models for existing software (e.g., Linux kernel) are large and need to be better managed	• Can relatively easily define examples and material about μTVL, CL and PSL for dissemination • Developing formal underpinnings for next generation feature modelling languages is within the expertise of the consortium	• Disseminate HATS work to companies using and planning to use the product line engineering approach • Build large case studies as demonstrators

Table 4.4: SWOT Analysis *Feature Modelling Framework* (continued)

Threats	**Strengths × Threats**	**Weaknesses × Threats**
• Results depend heavily upon technologies developed by others • A wealth of feature modelling languages already exist	• Rely on formal underpinnings and background knowledge to more effectively solve problems around feature models	• Make contact with companies and more applied researchers using product line engineering approaches

4.7 Delta Modelling Framework

Description

The Delta Modelling Framework provides a delta modelling design notation and programming language extension.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users. The concepts are suitable for usage in undergraduate and graduate education.

Application Domains

There is no specific application domain intended, usage in a broader sense is possible.

Usage Scenarios

The Delta Modelling Framework is designed to be used in the design and implementation phases of software product line engineering.

To support the design of software product lines, fragments of both design models and code can be specified using deltas. These can correspond to features, reusable chunks used to construct features, or to code repairing conflicts resulting from feature composition.

Delta modelling also supports the implementation of such fragments to incorporate into deployed products.

Final Status

The item will be available as a prototype for external usage.

Planned Steps Towards Exploitation

The Delta Modelling Framework has been incorporated into the ABS tool suite, so it will be exploited following the exploitation strategy of Exploitable Item 4.2.

Intellectual Property Protection

Currently there is no intellectual property protection planned.

Owner and Other Partners

CWI, UIO, UKL, NR, BOL, KUL, TUD

Main Contact

Reiner Hähnle {haehnle@cs.tu-darmstadt.de} (TUD), Dave Clarke {dave.clarke@cs.kuleuven.be} (KUL)

Related HATS Deliverables

D2.2

SWOT Analysis

See Table 4.5

Table 4.5: SWOT Analysis *Delta Modelling Framework*

	Strengths	**Weaknesses**
	• Expertise in programming language design and their implementation	• Limited experience with large scale software product line development • Limited contact with companies using product lines
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• Software product line engineering is increasingly becoming a technique for companies to consider • Good programming language support for product lines is lacking	• Can provide top quality prototypes and tutorial material for disseminating our results, attract users of the toolset	• Establishing contact with companies with the aid of the unique portfolio of tools we will offer, by presenting them at appropriate conferences and so forth
Threats	**Strengths × Threats**	**Weaknesses × Threats**
• Companies are reluctant to adopt new programming languages	• We can develop solid case studies to convincingly demonstrate the potential competitive advantage of using delta modelling	• Make contact with companies and more applied researchers using product line engineering approaches

4.8 Delta Modelling Development Workflow

Description

Workflow for guiding the development of software product lines using the Delta Modelling framework. By following the workflow, the resulting product lines will be unambiguous and feature complete.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users. The concepts are suitable for usage in undergraduate and graduate education.

Application Domains

There is no specific application domain intended, usage in a broader sense is possible.

Usage Scenarios

The Delta Modelling Development Workflow is designed to be used in the design and implementation phases of software product line engineering.

Final Status

The item will be available as a prototype for internal usage.

Planned Steps Towards Exploitation

An extended formalism is planned to describe the underlying technique, which better proves desirable properties.

Intellectual Property Protection

Currently there is no intellectual property protection planned.

Owner and Other Partners

CWI, UIO, UKL, NR, BOL, KUL, TUD

Main Contact

Reiner Hähnle {haehnle@cs.tu-darmstadt.de} (TUD), Dave Clarke {dave.clarke@cs.kuleuven.be} (KUL)

Related HATS Deliverables

D2.2

SWOT Analysis

See Table 4.6

Table 4.6: SWOT Analysis *Delta Modelling Development Workflow*

	Strengths	**Weaknesses**
	• The consortium has people experienced in setting up software product lines • The consortium has people experienced in formal methods • The consortium has several case studies which can be used to test the workflow	• The consortium does not have the resources or connections to set up a real industrial software product line from scratch in order to test the workflow in a real world application
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• Industry can benefit from new techniques to reuse code and develop software concurrently with different people in order to make optimal use of resources	• We can introduce the workflow in industry and demonstrate the opportunities it opens • We can back our claims with mathematically rigorous formal proofs as well as experience	• Find companies that are on the verge of writing software that is likely to evolve into a software product line

Table 4.6: SWOT Analysis *Delta Modelling Development Workflow* (continued)

Threats	**Strengths × Threats**	**Weaknesses × Threats**
• A lot of software product lines begin as traditional software systems rather than as product lines, so it is rare that the workflow can be used as intended, to build product lines from scratch	• Industry can be convinced of the cost-saving opportunities of planning ahead for a product line, rather than writing the software and then adapting it	• Demonstrate the workflow through tutorials and courses in order to demonstrate its benefits

4.9 Models and Constructs for Components

Description

Complex software systems, in particular distributed systems, are often being thought of and designed as structured composition of computational units referred to as components. These components are supposed to interact with each other following some predefined patterns or protocols. The notion of component is widely used in industry. This exploitable item will provide a calculus that extends the ABS language with a notion of component. Components can be seen as boxes containing and grouping objects, thus allowing the architectural structure of the system to be explicitly defined at runtime. A component can include, for instance, objects sharing some computational resources, or sharing a physical location, or sharing a security domain. Components can also be useful for adaptability and evolvability: components can be aggregated to form new components, components can move, components can be wrapped within other components that provide different functionalities to clients.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users.

Application Domains

There is no specific application domain intended, usage in a broader sense is possible.

Usage Scenarios

Specification and modelling of software systems is a typical usage of the constructs. The models for components can be used to give an explicit description of the architecture of the system at runtime. This is particularly useful for adaptable applications where the structure can be changed dynamically, thus allowing the update of the program during its execution, or its adaptation to modifications in its environment.

In addition, it will be used at the university in software engineering and formal methods courses at graduate and postgraduate level.

Final Status

The item will be available as a prototype for proving concepts.

Planned Steps Towards Exploitation

The component model was implemented in the ABS runtime. We are currently looking to include it in a type system and merge it with deployment components.

Intellectual Property Protection

Currently there is no intellectual property protection planned.

Owner and Other Partners

BOL, UKL

Main Contact

Davide Sangiorgi {davide.sangiorgi@cs.unibo.it} (BOL)

Related HATS Deliverables

D2.1, D3.1.a, D3.1.b

SWOT Analysis

See Table 4.7

Table 4.7: SWOT Analysis *Models and Constructs for Components*

	Strengths	**Weaknesses**
	• The addition of a component structure to ABS addresses a practical problem, as components are widely used in industry (at least at the architectural level) • Our components are formally defined • They offer a better notion of composition than objects, and allow some run-time adaptation of software • The structure of components helps to specify and prove properties about program behaviours	• The HATS consortium does not have the manpower to carry out a lot of experiments and case studies involving components
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• The concept of components is widely used in software systems, well beyond the application areas of HATS (e.g., software families). Finding appropriate models and reasoning techniques for components is highly significant	• Show that the combination of objects and components can be achieved in the setting of ABS using relatively simple linguistic constructs. This may push external people to investigate the concepts of components developed in other settings	• Try to push the notions and concepts developed into other projects, or in other collaborations with industries
Threats	**Strengths × Threats**	**Weaknesses × Threats**
• Other groups in the world are working on formal models for components, so the competition is strong • Industry requires tools and automated analysis, which, on the specific issue of components, might be difficult to attain within HATS	• Show the novelties and the good properties of our model to the other groups working on formal models for components and shape collaborations • Perform experiments to evaluate our component model in the context of the HATS case studies and demonstrations	• Disseminate the work via courses and tutorials to get other people involved

4.10 Behavioural Specification Framework for OO Components

Description

A behavioural specification framework for OO components, based on message traces of component interactions. The framework describes possibilities for specifying the behaviour of ABS components.

Intended Market or Sector

In general the research community in that area can make use of the framework. The concepts are suitable for usage in graduate and postgraduate education.

Application Domains

There is no specific application domain intended, usage in a broader sense is possible.

Usage Scenarios

In the design phase of software development the framework is used to provide a multi-level view of the behaviour of components.

Final Status

There will be a fine grained concept available that is ready for being applied.

Planned Steps Towards Exploitation

Development of a specification language and tools for concurrent components to be used in prototypical usage scenarios for industry and education.

Intellectual Property Protection

Currently there is no intellectual property protection planned.

Owner and Other Partners

UKL, CWI

Main Contact

Arnd Poetzsch-Heffter {poetzsch@cs.uni-kl.de} (UKL)

Related HATS Deliverables

D1.2, D2.5, D2.6

SWOT Analysis

See Table 4.8

Table 4.8: SWOT Analysis *Behavioural Specification Framework for OO-Components*

	Strengths	**Weaknesses**
	• Consortium members have know-how in the specification and verification of object-oriented components and programs • The HATS case studies are component-based case studies	• The specification framework is not known to the research community
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• There is a demand for specifying the behaviour of components in the design phase of software development	• Use of the case studies to demonstrate the advantages of the approach	• Disseminate and explain the framework to the research community
Threats	**Strengths × Threats**	**Weaknesses × Threats**
• The research community does not recognize the need for the specification framework	• Provide benchmarks to show the benefits of the specification approach	• Provide examples and motivation

4.11 HATS Methodology

Description

A software product line engineering methodology that integrates formal methods for modelling changes of software systems in terms of variability and evolution, while preserving trustworthiness properties.

Intended Market or Sector

Organizations that develop long-lived software systems and either focus on reusability and correctness (security, functional, resource) of long-lived software components or are interested to do so are potential users of the methodology.

Application Domains

Any long-lived software system that needs to be trustworthy and adaptable can be addressed by the methodology.

Usage Scenarios

By following the HATS methodology, organizations may apply formal techniques and tools contributed by the HATS project to develop their software systems into a software product line that increases reusability as well as correctness. Organizations may already have a software product line, in which case the HATS methodology helps increases reusability and correctness of the existing product line.

It is also possible to use the methodology in the Product Line Engineering courses at universities.

Final Status

The item will be available as a prototype for proving concepts.

Planned steps towards exploitation

Improving the methodology by:

- Describing stakeholder in each activity
- Describing input and output artefacts in each activity in the methodology
- Describing in detail the procedure of each activity in the methodology that is relevant to the HATS project
- Applying the methodology in a realistic software product line as part of the ongoing validation process.

Intellectual Property Protection

Currently there is no intellectual property protection planned.

Owner and Other Partners

All HATS Consortium Members are involved in the development of the methodology.

Main Contact

Peter Wong {peter.wong@fredhopper.com} (FRH)

Related HATS Deliverables

D1.1b, D1.5

SWOT Analysis

See Table 4.9

Table 4.9: SWOT Analysis *HATS Methodology*

	Strengths	**Weaknesses**
	• Know-how in the methodologies for object-oriented, concurrent, and distributed systems as well as product line engineering • Based on proven methodologies from industry • Backed by domain experts in product line engineering • Tool-oriented • Focuses on maintenance (evolvability) • Explicit and clear mapping from technologies delivered from consortium members to steps in the methodology • Iterative, concurrent, incremental • Allows partial adoption	• Lacks empirical evidence to support the practicality of the methodology • The HATS consortium does not have the manpower to implement industrial-strength tools • The documentation is designed for researchers
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• There is a demand for adaptable and trustworthy software • There is a demand for model-based software development • There is a demand for a methodology for developing and maintaining adaptable and trustworthy software • To the best of our knowledge there is no comparable methodology	• Demonstrations of the HATS methodology to companies that are interested in adaptable and trustworthy software, model-based development, and/or product line engineering • Use of the case studies for demonstration • Provide tutorial and practical guides so that interested companies to adopt the HATS methodology as soon and as easy as possible • Exercise every step in the methodology to demonstrate its benefit	• Disseminate the HATS methodology to industry • Emphasise that the HATS methodology is based on existing industrially-proven product line engineering methods

Table 4.9: SWOT Analysis *HATS Methodology* (continued)

Threats	**Strengths × Threats**	**Weaknesses × Threats**
• The work depends on external funding • The industry does not recognise the benefits of a methodology with emphasis on formal approaches • The cost-benefit relation must be positive • The industry requires high-quality tools • The industry may be reluctant to adopt a new development method • The industry needs to either hire new employees or train existing ones to adapt formal approaches/model based developments	• Provide benchmarks to show the benefits of the HATS methodology • Provide empirical evidence of the cost-benefit relation	• Provide documentation (guide books, manual) for end-users so that companies can adopt the methodology more easily • Search for industrial partners that are willing to adopt the methodology as a pilot programme • Search for partners that can implement industry-strength tools and/or make the software open-source to benefit from external developers

4.12 Integrated Development Environment for ABS

Description

An Eclipse-based IDE for the ABS language and integration of further HATS tools.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users. The concepts are suitable for usage in education.

Application Domains

Any software system that is fully or partially modelled with ABS can make use of this item.

Usage Scenarios

- User-friendly development of ABS model and easy access to the ABS tool chain
- Usage of the IDE in student courses.

Final Status

The item will be available as a prototype for proving concepts.

Planned Steps Towards Exploitation

- Improvement of the maturity of the tool chain in further projects
- Integration of more HATS related tools.

Intellectual Property Protection

BSD 3-Clause New or Revised License

Owner and Other Partners

UKL, UIO

Main Contact

Arnd Poetzsch-Heffter {poetzsch@cs.uni-kl.de} (UKL)

Related HATS Deliverables

D1.1, D1.2, D1.3, D1.4, D1.5

SWOT Analysis

See Table 4.2 (common analysis for exploitable items 4.2, 4.12, 4.14)

4.13 Formal Verification Tool for HATS ABS

Description

An incarnation of the state-of-art formal verification system KeY that has the HATS ABS language as its target. Hence, it makes it possible to formally verify ABS programs against their contracts.

Intended Market or Sector

Software engineering companies that develop safety-critical software would benefit from the item. The concepts are suitable for usage in graduate and postgraduate education.

Application Domains

Any software system that is fully or partially modelled with ABS can make use of this item. Especially safety-critical software is in focus.

Usage Scenarios

Central library methods in safety-critical systems might warrant the effort of formal specifications verification. The tool can support formal verification of safety-critical software that needs to be certified. In addition, it will be used at the university in graduate and postgraduate courses concerning formal verification.

Final Status

The item will be available as a prototype for internal usage.

Planned Steps Towards Exploitation

During project time: Finalizing implementation and evaluation along examples from the case study for D5.4. For post-project exploitation we will improve the automation, maintain the system and strive towards a deep integration into the HATS core tool suite. Usage in teaching.

Intellectual Property Protection

The tool is expected to be released under GPLv2.

Owner and Other Partners

TUD (owner), UIO (owner), IoC

Main Contact

Reiner Hähnle {haehnle@cs.tu-darmstadt.de} (TUD)

Related HATS Deliverables

D1.3, D2.5, D4.3

SWOT Analysis

See Table 4.10 (common analysis for exploitable items 4.4, 4.5, 4.13, 4.17, 4.23)

Table 4.10: SWOT Analysis *Formal Verification Tool for HATS ABS*

	Strengths	**Weaknesses**
	Consortium: • Several members do world-leading research on formal analysis and verification of programming languages • Know-how and experience in incremental and compositional verification • Considerable experience in creating and maintaining analysis and verification tools; specifically, KeY and COSTA are mature and well-established tools for Java which is a compilation backend of ABS • Know-how in a wide range of analysis and verification technologies *Tool:* • There is an existing implementation for symbolic execution and formal verification of CREOL programs within the KeY system. CREOL is closely related to Core ABS • The COSTA framework based on compilation and partial evaluation is very flexible in supporting different target programming languages • A case study for verification of product families (in Java) has been performed; further case studies in ABS are available	• It might be difficult to achieve a convincing degree of automation and usability during the duration of the project • It might be difficult to transfer the case studies performed for scientific publications to an industrial context

Table 4.10: SWOT Analysis *Formal Verification Tool for HATS ABS* (continued)

Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• There is a demand for advanced analysis and verification tools, in particular, for tools that work automatically • Reduced time-to-market is a powerful motivation to employ advanced verification tools • There is a growing demand for formal analysis in certification of safety-critical applications • To the best of our knowledge there is no symbolic execution engine for an advanced concurrent language such as ABS • New technologies (in the context of verification) such as delta-slicing of proofs or online partial evaluation have the chance to boost performance	• Demonstrate with case studies to interested companies that an extensive and reasonably automated formal analysis of ABS models of complex software is feasible	• Convince industrial stakeholders, foremost in the safety-critical systems area, to develop the HATS prototypes further in a follow-up project of more applied nature
Threats	**Strengths × Threats**	**Weaknesses × Threats**
• Symbolic execution and verification of a concurrent language of the complexity of ABS has not been done before • Some incremental techniques, such as delta-slicing of proofs have not been tried before • Additional funding required to produce industrial-strength tools cannot be obtained	• Use diverse methods and technologies in order to find out what works best • If full functional verification with a high degree of automation is not feasible, then concentrate on approximative methods which are still very useful in practice	• Concentrate on the techniques that maximize automation and usability, for example, test generation and visualization, and search for external partners with whom these can be developed to maturity

4.14 Interactive Simulator for ABS

Description

An interactive simulator/debugger for ABS based on the ABS Java backend.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users. The concepts are suitable for usage in education.

Application Domains

Any software system that is fully or partially modelled with ABS can make use of this item.

Usage Scenarios

- Understanding of the ABS concurrency and execution model
- Explanation and localization of faults (bugs) in ABS models
- Replay of scheduling histories
- Automatic generation of UML sequence diagrams
- Interactive stepping through an ABS model with different scheduling strategies
- Visualization of the heap, stack, and task states

Final Status

The item will be available as a prototype for proving concepts.

Planned Steps Towards Exploitation

- Application to larger case studies
- Application of the tool in prototypical usage scenarios for industry and education

Intellectual Property Protection

BSD 3-Clause New or Revised License

Owner and Other Partners

UKL

Main Contact

Arnd Poetzsch-Heffter {poetzsch@cs.uni-kl.de} (UKL)

Related HATS Deliverables

D1.1-D1.5, D2.3

SWOT Analysis

See Table 4.2 (common analysis for exploitable items 4.2, 4.12, 4.14)

4.15 Partial Evaluation-Based Test Case Generator

Description

A tool for automatic generation of test cases from Java bytecode programs or ABS models by relying on the technique of partial evaluation.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users.

Application Domains

Any software system that is written in Java (bytecode) or modelled in ABS might be the target of this item.

Usage Scenarios

The tool can be used to automatically generate test cases with respect to several coverage criterion, for example, loops must be executed at most n times, each instruction must be executed at least once, etc. These test cases are important for the validation of the developed models and software.

Final Status

The item will be available as a prototype for internal usage.

Planned Steps Towards Exploitation

Improve the tool to handle some of the cases studies used in HATS.

Intellectual Property Protection

The tool is expected to be released under the GPLv3.

Owner and Other Partners

UPM

Main Contact

Germán Puebla {`german@fi.upm.es`} (UPM)

Related HATS Deliverables

D2.3

SWOT Analysis

See Table 4.11

Table 4.11: SWOT Analysis *Partial Evaluation-based Test Case Generator*

	Strengths	**Weaknesses**
	• The consortium includes world leaders in glass-box test case generation for OO programs • The source code of PET, which is a test case generation tool for Java bytecode developed by members of the consortium, can be taken as a starting point for implementing a test case generation tool for (the sequential part of) ABS • HATS ABS has a formal semantics which allows performing test case generation directly on ABS models • ABS has a relatively simple concurrency model	• Test case generation of concurrent programs is challenging, because it is, among other things, difficult to enforce a particular schedule when re-executing • The available manpower for developing the prototype test case generation tool for ABS is limited
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• There is a growing industrial interest in automated test case generation which satisfy certain coverage criteria • Most of competing tools and techniques for test case generation focus on black-box approaches—glass-box approaches, if successful, can provide higher guarantees on the quality of the model/program under test	• The use of ABS allows relatively simple representations of realistic applications; a successful test case generation of such models can convince industry of the interest of the approach • Given the expertise of the consortium and the availability of a test case generation tool for Java bytecode, it should be possible to release a test case generation tool for (concurrent) ABS models before comparable systems for related programming languages are available	• Test case generation of concurrent applications should be performed fully automatically; this should definitely raise industrial interest in the system • Reusing parts of the existing test case generation tool for Java, together with the integration of the tool in the common user interface based on Eclipse should result in a sufficiently mature system to show the industrial interest of the tool

Table 4.11: SWOT Analysis *Partial Evaluation-based Test Case Generator* (continued)

Threats	**Strengths × Threats**	**Weaknesses × Threats**
• Industry may find the final prototype not usable enough • Competing systems may be released	• Within the manpower available within the project, exploit the fact that a test case generation tool is available and that the level of expertise is high in order to produce a prototype as mature as possible • Aim at releasing the prototype test case generation tool soon, so as to avoid that competing systems are released before	• Focus research activity in handling concurrency, which is the main open question • The results of the analyzer should be of practical interest to industry

4.16 Monitor Inlining Tool

Description

A tool to embed program monitors for testing and security monitoring into Java programs.

Intended Market or Sector

Software development companies and companies who have security demands on their software applications are potential users of the tool. The tool is also likely to be used in software security education.

Application Domains

There is no specific application domain intended, usage in a broader sense is possible.

Usage Scenarios

The inline monitors can be used to support bug identification during software development and to monitor security in running software

Final Status

The item will be available as a prototype for proving concepts.

Planned Steps Towards Exploitation

The monitor inlining tool, including its recent extension to Android, will be exploited in a new undergraduate course on software safety and security at KTH which was given first time this year. In addition, the inlining tool is likely to be further developed as part of a new project on secure virtualization which was started recently at KTH.

Intellectual Property Protection

It is intended to use LGPL.

Owner and Other Partners

KTH

Main Contact

Mads Dam {mfd@kth.se} (KTH)

Related HATS Deliverables

D3.4

SWOT Analysis

See Table 4.12

Table 4.12: SWOT Analysis *Monitor Inlining Tool*

	Strengths	**Weaknesses**
	• Expertise in security policy specification, enforcement and verification • Expertise in the development of tools for formal verification of software • Case studies are being developed.	• Insufficient resources to bring the technology and tools to a mature industrial level • The texts and documentation produced usually targets scientists and not industry
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• Growing demand for trustworthy software • Adoption of security policy enforcement in software development • Growing demand for formal modelling techniques for software security	• The design of convincing prototype tools • Performing industrial case studies	• Producing convincing proof-of-concept tools and case studies
Threats	**Strengths × Threats**	**Weaknesses × Threats**
• Industry does not always recognize the benefits of adopting formal design techniques and tools • Industry expects high-quality mature tools for which the time frame and funding of the project is insufficient	• Demonstration of tools and techniques • Choice of convincing case studies	• Use of rapid prototyping to develop user-friendly tools at the possible expense of efficiency • Producing short but usable user manuals and testing these on “naive” users such as masters students

4.17 Visual Debugging Tool for HATS ABS

Description

The KeY Symbolic Execution Debugger for ABS is a debugging tool that allows the visual representation of all symbolic execution paths of a HATS Core ABS model.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users. The concepts are suitable for usage in graduate and postgraduate education.

Application Domains

Any software system that is fully or partially modelled with ABS can make use of this item.

Usage Scenarios

Code understanding, explanation and localization of faults (bugs) in ABS models. The visualization capabilities go far beyond, for example, sequence diagrams, because all computation paths are (symbolically) represented. It is also possible to visualize the heap structure. This is particularly useful in early design and prototyping, because no concrete initial state is required to be specified in order to start debugging. The tool can support formal development of safety-critical software that needs to be certified. In addition, it will be used at the university in graduate and postgraduate courses concerning formal verification and automated debugging.

Final Status

The item will be available as a prototype for proving concepts.

Planned Steps Towards Exploitation

- Connect with SE extraction currently implemented for Java Backend
- Evaluate on examples

Intellectual Property Protection

The tool is expected to be released under the GPLv3.

Owner and Other Partners

TUD

Main Contact

Reiner Hähnle {haehnle@cs.tu-darmstadt.de} (TUD)

Related HATS Deliverables

D1.3, D2.3

SWOT Analysis

See Table 4.10 (common analysis for exploitable items 4.4, 4.5, 4.13, 4.17, 4.23)

4.18 Learning-Based Test Tool

Description

A platform for black box requirements testing of executable ABS models using linear temporal logic requirements.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users. The concepts are suitable for usage in education.

Application Domains

Reactive and embedded systems, e.g. client-server configurations.

Usage Scenarios

It can be used at levels of unit, integration, and system testing. For scalability reasons, it is more efficient at unit level.

Precondition: The user has an ABS model, and one or more propositional linear temporal logic requirements. These could be safety or liveness requirements. The user creates an executable Java program using the ABS Java backend.

The user creates a data interface to symbolically encode key data values as Boolean vectors for the system. This is stored as a lookup table. The user chooses a test termination criterion, such as a time bound. Some other test parameters need to be chosen (tuning). The test tool runs in background.

Post Condition. The test tool terminates according to criterion. For each user requirement a pass/fail/warning verdict is delivered. In the case of warning or fail, one or more failed test cases are returned for software autopsy using a debugger. A graphical model of the inferred system model is returned in .dot format, for coverage analysis.

Final Status

The item will be available as a prototype for external usage.

Planned Steps Towards Exploitation

A pre-release of the tool LBTest 0.9 will be available in autumn 2012. This release will be made available within the ARTEMIS project MBAT (Combined Model-based Analysis and Testing) for industrial partners to evaluate the tool. KTH will also perform MBAT case studies on the tool. LBTest 1.0 will be made available for download on the KTH web site during winter 2012.

Intellectual Property Protection

To be discussed with KTH administration.

Owner and Other Partners

KTH

Main Contact

Karl Meinke {karlm@nada.kth.se} (KTH)

Related HATS Deliverables

D2.3

4.19 Software Product Line Verification Tool

Description

The tool allows software product lines described with hierarchical variability models to be model checked against temporal safety properties of sequences of method invocations formalized in temporal logic. Scalability is achieved through relativisation of the global correctness property on properties of the variation points of the variability model. As input to the tool serves an ABS model or a Java bytecode program, decorated with annotations (pragmas) capturing the variability model, and with annotations for all global and local specifications. The tool reports success, or else identifies the components that have to be corrected.

Intended Market or Sector

Software engineering companies that have software product line development.

Application Domains

Software for which it is critical how it accesses resources over time, such as correct API usage, is in focus.

Usage Scenarios

Companies who develop SPL can use the tool to assure that no product of the SPL violates given temporal safety properties, such as API usage protocols or security policies of the underlying execution platform.

Final Status

The item will be available as a prototype for proving concepts.

Planned steps towards exploitation

The tool will be combined with another tool for hierarchical variability model mining from already developed sets of software artifacts. The extracted variability models will serve as an input to the verification tool and guide the annotation process.

Intellectual Property Protection

It is intended to use LGPL.

Owner and Other Partners

KTH

Main Contact

Dilian Gurov {dilian@csc.kth.se} (KTH)

Related HATS Deliverables

D2.5, D4.3

SWOT Analysis

See Table 4.13

Table 4.13: SWOT Analysis *Software Product Line Verification Tool*

	Strengths	**Weaknesses**
	• Expertise in formal modelling and development of Software Product Lines • Expertise in components and compositional verification • The ABS language is designed to support verification • Expertise in the development of tools for formal verification of software • Case studies are being developed	• Insufficient resources to bring the technology and tools to a mature industrial level • The texts and documentation produced usually targets scientists and not industry
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• Growing demand for trustworthy software • Adoption of the Software Product Lines paradigm in software development • Growing demand for formal modelling techniques for Software Product Lines	• The design of convincing prototype tools • Performing industrial case studies	• Producing convincing proof-of-concept tools and case studies
Threats	**Strengths × Threats**	**Weaknesses × Threats**
• Industry does not always recognize the benefits of adopting formal design techniques and tools • The cost/benefit ratio has to be favourable • Industry expects high-quality mature tools for which the time frame and funding of the project are insufficient	• Demonstration of tools and techniques • Choice of convincing case studies	• Use of rapid prototyping to develop user-friendly tools at the possible expense of efficiency • Producing short but usable user manuals and testing these on “naive” users such as masters students

4.20 ABS Product Configurator

Description

Tool that supports the configuration of products (selection of features) from a product line modeled in the ABS framework. It addresses not only variability in functional features, but also variability in quality, specifically performance and security. The core features of the configurator are:

- It takes into consideration the constraints specified by the user (for example: required features and cost constraints)
- It uses heuristics based on performance and security annotations to suggest suitable configurations
- It provides the quality ratings of the suggested configurations.

Intended Market or Sector

Software engineering companies that have software product line development. All companies that need to configure products from complex product lines modeled in ABS and want to ensure product performance and/or security are intended customers.

Application Domains

There is no specific application domain intended, usage in a broader sense is possible.

Usage Scenarios

Developers can use it to configure the desired products from product lines modeled using the HATS ABS framework. This prototype tool can be used in the context of performance and security aware configuration and can be extended with more quality characteristics.

Final Status

The item will be available as a prototype for internal usage.

Planned Steps Towards Exploitation

- Investigating the use of a SMT solver
- Developing more user friendly interfaces
- Making the source open to the community
- Visualizing configurations with detailed quality ratings.

Intellectual Property Protection

The tool is expected to be released under the GPLv3.

Owner and Other Partners

FRG (configurator), UPM (performance annotator), and NR (security knowledge). KUL is also a partner in this exploitable item.

Main Contact

Karina Villela {karina.villela@iese.fraunhofer.de} (FRG)

Related HATS Deliverables

D4.4

SWOT Analysis

See Table 4.14

Table 4.14: SWOT Analysis *ABS Product Configurator*

	Strengths	**Weaknesses**
	• Partners: Know-how in product line engineering, static analysis and security • Configurator: addresses quality variability • Development Environment: Possibility of integration in the ABS Eclipse platform	• Imprecise quality annotations are used as part of the heuristics • Performance of every feature is analysed statically and in isolation • CSP solver is used which does not scale much • Not enough manpower to implement an industry-strength tool
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• There is a significant demand for configuring software products from product lines • There is a significant demand for a configurator that addresses quality variability • No comparable configurator is available	• Demonstrating the ABS Product Configurator integrated into the ABS IDE	• Runtime monitoring can improve the accuracy of the performance annotations
Threats	**Strengths × Threats**	**Weaknesses × Threats**
• Very specific configurator, if attached to the HATS ABS framework • Benefit provided by the configurator may not be clear • Industry demands integrated and high quality tools	• Making the configurator independent from the specific language used for variability modeling • Demonstrating the benefits of ABS modeling and configuration support	• Investigating the use of a SMT solver in the HATS framework instead of a CSP solver in order to address potential scalability issues • Finding partners interested in further developing the configurator

4.21 Static Analysis Techniques for Deadlock Freedom

Description

The ABS language uses object groups to deal with concurrency. This concept can be used to ensure consistency among the concurrent processes that try to access and modify some shared data, but also can cause unexpected deadlocks. We have thus developed two techniques for the deadlock analysis:

- a technique based on contracts, which are basically types that give an abstract description of the behaviour of the object methods tailored at deadlock analysis
- a technique based on data abstraction and translation into Petri Nets, which are then analysed for reachability of deadlock marking.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users.

Application Domains

There is no specific application domain intended, usage in a broader sense is possible.

Usage Scenarios

Analysis of software systems. The proposed technique can be used to ensure that software systems are deadlock free. The technique can be applied to systems modeled in ABS and can be extended to deal with other Java-like languages and to languages for service oriented computing.

In addition, it will be used at the university in software engineering and formal methods courses at graduate and postgraduate level.

Final Status

There will be a fine grained concept available that is ready for being applied.

Planned Steps Towards Exploitation

The development of the tool is on-going and it will continue after the end of the project.

Intellectual Property Protection

Currently there is no intellectual property protection planned.

Owner and Other Partners

BOL

Main Contact

Davide Sangiorgi {davide.sangiorgi@cs.unibo.it} (BOL)

Related HATS Deliverables

D2.4, D2.5

SWOT Analysis

See Table 4.15

Table 4.15: SWOT Analysis *Type Systems for Deadlock Freedom*

	Strengths	**Weaknesses**
	• The language on which the analysis is carried out is developed within HATS, and the HATS consortium has the expertise on object-oriented languages, static analysis, and type systems for concurrency that is required for carrying out the analysis	• The HATS consortium does not have the manpower to carry out large experiments and case studies on this specific item
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• Deadlock is one of the most interesting and practically relevant problems in concurrency	• Show, by means of examples, that the technique developed can indeed allow one to detect non-trivial and subtle forms of deadlock	• Try to get other people involved in the problem; possibly invest a PhD thesis on the problem
Threats	**Strengths × Threats**	**Weaknesses × Threats**
• A lot of work in the literature exists on deadlock. External people have to be convinced that the technique developed is interesting even outside the ABS language	• Excellence in dissemination (e.g., publish the work in some prestigious conference)	• Develop some automated, or semi-automated, analysis tools, based on the techniques and methods investigated

4.22 Cost and Termination Analyzer

Description

A cost and termination analyser for ABS models and Java bytecode programs.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users. The concepts are suitable for usage in graduate education.

Application Domains

Any software system that is written in Java (bytecode) or modeled with HATS Core ABS can be analysed with the tool.

Usage Scenarios

The system can be used to verify that Java (bytecode) programs or ABS models meet their expected resource consumption, for example, the number of execution steps (or derivations), the total memory consumption, the peak memory consumption when considering a specific garbage collection scheme, the number of tasks that might run in parallel, etc. It is also possible to formally verify the correctness of the inferred bounds using the KeY system (see exploitable item 4.23).

The tool can be used as a support tool for teaching complexity of algorithms. In particular, observing the intermediate steps applied by the system helps in understanding how to use classical techniques to infer upper bounds.

Final Status

The item will be available as a prototype for internal usage.

Planned Steps Towards Exploitation

Improve the different aspects of the prototype in order to be able to analyze some of the case studies used in HATS.

Intellectual Property Protection

It is intended to use the GNU General Public License.

Owner and Other Partners

UPM

Main Contact

Germán Puebla {`german@fi.upm.es`} (UPM)

Related HATS Deliverables

D4.2

SWOT Analysis

See Table 4.16

Table 4.16: SWOT Analysis *Cost and Termination Analyzer*

	Strengths	**Weaknesses**
	• The consortium includes world leaders in static resource (cost) analysis of OO programs • The source code of COSTA, which is a prototype cost analyzer for Java bytecode developed by members of the consortium, can be taken as a starting point for implementing an analyzer for the sequential part of ABS • ABS has a formal semantics which allows performing resource analysis directly on ABS models • ABS has a relatively simple concurrency model	• Static cost analysis of concurrent programs has not been performed before • The available manpower for developing the prototype analyzer for ABS is limited
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• There is a growing industrial interest in automated resource guarantees • To the best of our knowledge, no other competing systems are in a position to provide similar results. This is especially true for concurrent applications	• The use of ABS allows relatively simple representations of realistic applications; a successful resource analysis of such models can convince Industry of the interest of the approach • Given the expertise of the consortium and the availability of a resource analyzer for Java, it should be possible to release a cost analyzer for (concurrent) ABS models before comparable systems for related programming languages are available	• Resource analysis of concurrent applications should be performed fully automatically; this should definitely raise industrial interest in the system • Reusing parts of the existing resource analyzer for Java together with the integration of the tool in the common user interface based on Eclipse should result in a sufficiently mature system to show the industrial interest of the analyzer

Table 4.16: SWOT Analysis *Cost and Termination Analyzer* (continued)

Threats	**Strengths × Threats**	**Weaknesses × Threats**
• Industry may find the final prototype not usable enough • Competing systems may be released	• Within the manpower available within the project, exploit the fact that a resource analyzer is available and that the level of expertise is high in order to produce a prototype as mature as possible • Aim at releasing the prototype analyzer soon, so as to avoid that competing systems are released before	• Focus research activity in handling concurrency, which is the main open question • The results of the analyzer should be of practical interest to industry

4.23 Automated Derivation and Verification of Resource Bounds

Description

The state-of-art resource analysis system COSTA (see *Cost and Termination Analyzer* in Section 4.22) in its incarnation for core ABS and the formal verification tool KeY ABS (see *Formal Verification Tool for HATS ABS* in Section 4.13) are combined to KeY-COSTA to permit fully automatic derivation and formal verification of resource bounds of ABS models. This includes memory as well as runtime estimations.

Intended Market or Sector

Software engineering companies that develop software that needs to run in tight resource bounds would benefit from the item. The concepts are suitable for usage in graduate and postgraduate education.

Application Domains

Any software system that is modeled with HATS Core ABS and compiled to Java Code via the ABS backend.

Usage Scenarios

KeY-COSTA can be used to automatically derive symbolic expressions that give lower and upper bounds on runtime (number of executed steps) and memory of ABS models. The resulting bounds are formally verified and certificates are available. Interesting scenarios are devices with limited resources, but also models with real-time constraints. In addition, KeY-COSTA will be used at the university in graduate and postgraduate courses concerning complexity/termination of algorithms.

Final Status

The item will be available as a prototype for proving concepts.

Planned Steps Towards Exploitation

The tool can be used within Eclipse, and it is publicly available for download. Recently it has been improved to handle heap properties such as acyclicity, reachability, and sharing.

Intellectual Property Protection

The tool is expected to be released under the GPLv3.

Owner and Other Partners

CTH (for the KeY part), UPM (for the COSTA part)

Main Contact

Reiner Hähnle {haehnle@cs.tu-darmstadt.de} (TUD), German Puebla {german@fi.upm.es} (UPM)

Related HATS Deliverables

D1.3, D4.2

SWOT Analysis

See Table 4.10 (common analysis for exploitable items 4.4, 4.5, 4.13, 4.17, 4.23)

4.24 Timed Resource Consumption Analysis and Simulation

Description

An ABS language extension for expressing timed model behavior and resource consumption (execution cost), and a tool for executing and simulating timed resource-aware models. The main features are:

- A way of modeling *deadlines* for method calls and *durations* (elapsed time) during execution. Both deadlines and durations can be constant or depend on object state.
- A way of executing COGs on *deployment components* (entities supplying computing resources, roughly corresponding to physical servers) and specifying *execution costs* of statements.
- An implementation of timed, resource-aware model simulation and execution using the existing Maude backend.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users.

Application Domains

There is no specific application domain intended, usage in a broader sense is possible.

Usage Scenarios

Developers can use timed, resource-aware models to gain an understanding of deployment and performance issues before implementing the system. The ABS language and core tools and the ABS modelling workflows are used mostly as-is.

Final Status

The item will be available as a prototype for external usage.

Planned Steps Towards Exploitation

- Evaluate the tool against more case studies.
- Visualize results of performance simulations.

Intellectual Property Protection

The tool is expected to be released under the GPLv3.

Owner and Other Partners

UIO

Main Contact

Einar Johnsen {`einarj@ifi.uio.no`} (UIO)

Related HATS Deliverables

D2.1

SWOT Analysis

See Table 4.17

Table 4.17: SWOT Analysis *Timed Resource Consumption Analysis and Simulation*

	Strengths	**Weaknesses**
	• Strong tool support via re-using main ABS product • Understandable tool via formal semantics of resource consumption • Resource and timing aspects orthogonal to functional model, can be added easily	• Accuracy of results depends on quality of annotations
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• Increasing importance of formal cloud computing development	• Ability to model and simulate deployment scenarios using the HATS approach • Existing case studies can be augmented with resource and time models	• Existing case studies can be used to validate the approach

Table 4.17: SWOT Analysis *Timed Resource Consumption Analysis and Simulation* (continued)

Threats	**Strengths × Threats**	**Weaknesses × Threats**
• Approach is somewhat specific to the HATS ABS framework • Industry demands integrated and high quality tools	• Demonstrating the benefits of ABS modeling • Investigating how to integrate this approach with other methodologies	• Educating modellers on the use of the tools and modelling approach

4.25 Hybrid Analysis for Evolvability

Description

A suite of techniques for analysing evolution of software product lines in order to ensure the maintenance of trustworthiness as modifications are deployed.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users. The concepts are suitable for usage in graduate and postgraduate education.

Application Domains

Application domains where trustworthiness must be assured are in focus.

Usage Scenarios

Hybrid analysis for evolvability will be needed for the evolution and maintenance of software product lines.

When changes need to be made to a software product line, either resulting from maintenance or the addition of new functionality, appropriate static, dynamic and hybrid analyses need to be applied to ensure that performing the modification will preserve the trustworthiness of the entire product line.

Final Status

There will be a fine grained concept available that is ready for being applied.

Planned Steps Towards Exploitation

The results of this work will be incorporated into the frontend of the ABS as a library and as a special extension of the backend of the ABS language. These can be used by users of the ABS tool suite.

Intellectual Property Protection

Currently there is no intellectual property protection planned.

Owner and Other Partners

KUL (owner), UIO, BOL

Main Contact

Dave Clarke {dave.clarke@cs.kuleuven.be} (KUL)

Related HATS Deliverables

D3.3

SWOT Analysis

See Table 4.18

Table 4.18: SWOT Analysis *Hybrid Analysis for Evolvability*

	Strengths	**Weaknesses**
	• Solid expertise in formal analysis techniques	• Lack of extensive practical experience to evaluate real usefulness of developed approaches
Opportunities	**Strengths × Opportunities**	**Weaknesses × Opportunities**
• Good analysis tools will increase the quality of software product lines in situations where dynamicity and evolution occur	• Building good tools to support our formal analyses can help the techniques reach the appropriate markets	• Establishing contact with companies with the aid of the unique portfolio of tools we will offer, by presenting them at appropriate conferences and so forth
Threats	**Strengths × Threats**	**Weaknesses × Threats**
• It may not be theoretically possible to design useful analyses	• The combined expertise will be crucial for designing computationally feasible approaches	• Apply techniques to case studies as they are being designed to better understand their limitations

4.26 Model Mining Algorithms and Tool

Description

Algorithms and corresponding prototype tool for extracting models from software written in the Java programming language. The extracted, or mined, models are written in the ABS language.

The end result will be one prototype tool.

Intended Market or Sector

In general all companies that do their business with software engineering are potential users.

Application Domains

There is no specific application domain intended, usage in a broader sense is possible.

Usage Scenarios

These items are relevant in cases where legacy software is to be modelled using the ABS language, and where those doing the modelling need assistance. Such need for assistance could be due to inexperience with ABS.

Given software written in Java, the algorithms and tool could be used to extract models, or early versions of models, semi-automatically from the Java code. The developers of the Java code would have to be available to know what the relevant parts of the code are and in order to answer questions from the tool.

Final Status

The item will be available as a prototype for proving concepts.

Planned Steps Towards Exploitation

- Generalize mining techniques developed for one domain (distributed message systems) to distributed systems in general.
- Integrate it better with the ABS Eclipse platform.

Intellectual Property Protection

The algorithms and other scientific results will be published, and thus no legal constraints on their usage are expected.

The use of the tool will be limited only by the software license to be selected by the project.

Owner and Other Partners

NR, KTH, UPM

Main Contact

Bjarte M. Østvold {bjarte@nr.no} (NR)

Related HATS Deliverables

D3.2

SWOT Analysis

See Table 4.19

Table 4.19: SWOT Analysis *Model Mining Algorithms and Tool*

	Strengths	**Weaknesses**
	• Experience in fields relevant to the subject matter: machine learning, static analysis, ... • We can build on existing tools we have made and extend them • Real code is available for experimenting on in case studies, and the developers of the code are in the project	• Different research groups have quite different methods for attacking the mining problem • Producing a polished tool is not realistic given the available manpower • We do not currently have a larger example that is non-sequential and that has both Java and ABS code
Opportunities • Little if any existing work on mining and product lines: thus, low hanging fruits to be picked • The work task is intentionally loosely defined to allow for experimentation • Results that read and produce code get credibility from developers • Models are hard to write, so developers require assistance, something that mining techniques provide	**Strengths × Opportunities** • Extend our existing tools and experience to make a prototype tool that reads Java code and produces results that developers can relate to their code	**Weaknesses × Opportunities** • Make smaller examples that are concurrent, and work with others on larger examples of that kind • Make a tool that works well in special cases, to have something to show developers and assist them
Threats • Little if any existing work on mining and product lines: thus harder to motivate and publish • Software engineers do not recognize the need for mining techniques or even understand what they are • Underspecified problem makes it harder to know what a good result is	**Strengths × Threats** • Relate work on mining to our previous research using similar methods and solve related problems • Work with real code and developers in order to understand how to educate other developers	**Weaknesses × Threats** • Strive to relate the different methods for attaching the mining problem to address a parts in a whole • Make that work well in special cases, to lower risk of underspecified problem and make motivation easier

4.27 Product Adaptation Framework

Description

The Product Adaptation Framework is designed to properly handle adaptation at runtime in products derived from a product line. The runtime adaptation language is targeted at defining the exact adaptations that are possible to be done in the product at runtime without harming the feature model constraints (defined using μTVL). The goal is to guarantee a consistent product configuration when the product is adapted at runtime. This framework is enabled by an infrastructure that supports the application of deltas at runtime, which has been developed by KUL.

Intended Market or Sector

Customers of the HATS ABS framework are the potential customers for the Runtime Adaptation Framework. All companies that maintain high available systems or that spend too much effort and time on the deployment of adaptations of the same product are intended customers.

Application Domains

There is no specific application domain intended, usage in a broader sense is possible.

Usage Scenarios

A possible usage scenario is, for example, mobile carriers that need to support adaptation of their systems but at the same time cannot have the system down for a while in order to re-deploy. As not every change at runtime leads to a consistent configuration of the product, this item defines all possible adaptations to consistent states.

Final Status

The item will be available as a prototype for proving concepts.

Planned Steps Towards Exploitation

- Improve the framework
- Create training for companies for dealing with adaptable products
- Experiment with it in different contexts
- Apply the framework in a Product Line Engineering real context.

Intellectual Property Protection

The tool is expected to be released under the GPLv3.

Owner and Other Partners

FRG (runtime adaptation language) and KUL (infrastructure for applying deltas at runtime).

Main Contact

Karina Villela {karina.villela@iese.fraunhofer.de} (FRG)

Related HATS Deliverables

D3.5

SWOT Analysis

See Table 4.20

Table 4.20: SWOT Analysis *Product Adaptation Framework*

	Strengths *Partners:* • Know-how in SPLE and variation and adaptation management *Adaptation language:* • Formal semantics • Designed to support verification • Families of products have now a formal and concrete way to define possible adaptations	**Weaknesses** • The Runtime Adaptation Framework requires some kind of infrastructure that supports adaptation of the deployed products at runtime • Not enough manpower to implement an industry-strength framework
Opportunities • There is an increasing demand for highly adaptable software products • There is a demand for model-based software development	**Strengths × Opportunities** • Use of the HATS case studies to demonstrate the advantages of the approach	**Weaknesses × Opportunities** • Making the framework independent of the adaptation mechanism infrastructure • Finding partners to further develop the framework
Threats • Industry does not always recognize the benefits of adopting formal design techniques and tools	**Strengths × Threats** • Demonstrating the feasibility and benefits of the framework in practice	**Weaknesses × Threats** • Developing end-user documentation so that companies can more easily adopt the framework

Chapter 5

Demonstrator and Feedback from Demonstrations

To present the HATS approach and its tools to companies and researchers, we created a demonstrator. The demonstrator is updated continuously according to the current status of the HATS project. In its current state the demonstrator consists of:

1. A tutorial presentation that introduces the HATS approach in about 30 minutes. This presentation includes a general overview of the HATS approach, an introduction into the ABS language (including the four variability modeling sub-languages), an overview of the ABS tool suite, and a brief explanation of the HATS case studies.

2. The main parts of ABS tool suite is packaged into an Eclipse plugin. The tools can be directly executed on the operating systems Windows, Linux, and MacOSX, and only require an installed Java Runtime Environment in version 6 (JRE).

3. Representative ABS models for the three HATS case studies, namely the Trading System, the Replication Server, as well as the Virtual Office of the Future.

Two demonstrations of the HATS approach were given at Ericsson AB in Gothenburg, Sweden. At the first occasion (6 April 2011), we presented the HATS approach to more than 50 Ericsson developers and engineers, while the second meeting (22 September 2011) was with a smaller group. The reaction of the developers and group leaders at Ericsson was very positive, however, we have been told that the current policy of the senior management is to abstain from R&D efforts that take more than 3 months to realize. In general, the HATS approach made a good impression and we are hopeful that they will decide applying it in a more appropriate circumstance.

Another demonstration of the HATS approach was given on 11 November 2011 at SAP Research GmbH in Darmstadt to a group of research engineers and research managers. They showed considerable interest in the ABS technologies. The ongoing virtualization and cloud computing efforts at SAP demand advanced modeling capabilities, and the ABS technologies were seen as promising. So far, two concrete outcomes have arisen from the demonstration at SAP Research:

- A jointly (by TUD and SAP) supervised MSc Thesis with the title "Abstract Modeling of Business Software" done by Marko Martin. The results will be presented at the Annual Meeting in Valencia and will also be published in a suitable venue.

- An informal agreement to collaborate on a proposal for a follow-up project to HATS that uses the ABS technologies.

On 29 August 2012, a presentation was given at the Websand (`http://www.websand.eu`) project meeting. Websand is a FP7 EU project with the title "Server-driven Outbound Web-application Sandboxing", where security plays a major role.

Furthermore, a demonstration at Google Germany GmbH in Munich is arranged to take place on 16 September 2012.

Finally, the HATS approach and its tools will not only be presented to large companies, but also to small and medium-sized enterprises (SME). A demonstration is planned to members of an SME network of software companies (STI) in Rhineland-Palatinate (Germany) for October 2012.

Chapter 6

Conclusion

Exploitation of the results of a research project depends on three main factors: 1) how much those results contribute to solve a practical problem in the industry, 2) how easy the adoption of the project results in industrial settings is, and 3) what the solutions offered by competitors are. In this deliverable we have addressed those three factors, by performing a market analysis and by analysing the project results in terms of strengths, weaknesses, opportunities and threats. The latter is referred as SWOT analysis.

Two of the tools (*FaMa-Framework* and *SPLOT*) presented in Section 2.1.2 (*SPL Engineering Tools*) already integrate mature formal approaches into SPL Engineering. Another tool (*FeatureIDE*) supports delta-oriented programming [63]. However, each of these tools only addresses an individual aspect. The HATS outcome is not only a tool, but also a methodological framework that jointly address variability, evolvability, and trustworthiness. In fact, methodological support that guides the application of formal approaches in SPL Engineering processes has received few attention so far. The HATS consortium members have made a great effort towards providing such support, which includes capturing in the HATS methodology the interplay of the different technologies developed in the project. The automation of both engineering and quality assurance activities will reach an unprecedented level due to the replacement of informal processes with rigorous ones based on formal semantics.

In order to successfully exploit the project results, the HATS consortium members have tried to propose lightweight formal approaches that can be easily understood and integrated with the current practices, and that are scalable and modular, but of course there will still be place for improvement of the developed technologies after the end of the project. One example is the provision of a design schema capable of breaking a product line into modules, so that each module can cover certain aspects instead of having e.g. one big feature model file and one big product line configuration file for the whole product line. The HATS consortium members also have tried to offer integrated, standardized, and end-to-end tooling support. The steps towards the full integration of the developed tools will be provided in Deliverable 1.5, to be delivered by the end of the project (T48).

Bibliography

[1] M. Acher, P. Collet, P. Lahire, S. Moisan, and J.-P. Rigault. Modeling variability from requirements to runtime. In I. Perseil, K. Breitman, and R. Sterritt, editors, *Proc. of 16th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2011)*, pages 77–86, April 2011.

[2] G. H. Alférez and V. Pelechano. Context-aware autonomous web services in software product lines. In E. Almeida, T. Kishi, C. Schwanninger, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 1, pages 100–109. IEEE Computer Society, August 2011.

[3] M. Anastasopoulos and A. Balogh. Model-driven development of particle system families. In *Workshop on Model Based Methodologies for Pervasive and Embedded Systems*, pages 1–11. IEEE Press, 2007.

[4] P. Asirelli, M. ter Beek, S. Gnesi, and A. Fantechi. Formal description of variability in product families. In E. Almeida, T. Kishi, C. Schwanninger, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 1, pages 130–139. IEEE Computer Society, August 2011.

[5] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[6] E. Bagheri, T. Noia, A. Ragone, and D. Gasevicl. Configuring Software Product Line Feature Models Based on Stakeholders' Soft and Hard Requirements. In *Software Product Line Conference*, volume 1, pages 16–31, 2010.

[7] M. Becker, S. Kemmann, and K. Shashidhar. Integrating Software Safety and Product Line Engineering using Formal Methods: Challenges and Opportunities. In *Software Product Line Conference*, volume 2, pages 129–136, 2010.

[8] S. Becker, W. Hasselbring, A. Paul, M. Boskovic, H. Koziolek, J. Ploski, A. Dhama, H. Lipskoch, M. Rohr, D. Winteler, S. Giesecke, R. Meyer, M. Swaminathan, J. Happe, M. Muhle, and T. Warns. Trustworthy software systems: a discussion of basic concepts and terminology. *SIGSOFT Software Engineering Notes*, 31(6), 2006.

[9] P. Broek. Optimization of Product Instantiation using Integer Programming. In *Software Product Line Conference*, volume 2, pages 107–112, 2010.

[10] P. Broek, I. Galvao, and J. Noppen. Merging Feature Models. In *Software Product Line Conference*, volume 2, pages 83–90, 2010.

[11] E. Cavalcante, A. Almeida, T. Batista, N. Cacho, F. Lopes, F. C. Delicato, D. Souza, T. Sena, and P. F. Pires. Exploiting software product lines to develop cloud computing applications. In *16th International Software Product Line Conference*, September 2012.

[12] L. Chen and M. A. Babar. Variability Management in Software Product Lines: An Investigation of Contemporary Industrial Challenges. In *Software Product Line Conference*, volume 1, pages 166–180, 2010.

[13] S. Chen and M. Erwig. Optimizing the product derivation process. In E. Almeida, T. Kishi, C. Schwanninger, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 1, pages 35–44. IEEE Computer Society, August 2011.

[14] D. Clarke, N. Diakov, R. Hähnle, E. B. Johnsen, G. Puebla, B. Weitzel, and P. Y. H. Wong. HATS: A Formal Software Product Line Engineering Methodology. In *Proceedings of International Workshop on Formal Methods in Software Product Line Engineering*, Sept. 2010.

[15] D. Clarke and J. Proença. Towards a Theory of Views for Feature Models. In *Proceedings of International Workshop on Formal Methods in Software Product Line Engineering*, Sept. 2010.

[16] M. Cordy, P. Schobbens, P. Heymans, and A. Legay. Towards an incremental automata-based approach for software product-line model checking. In *16th International Software Product Line Conference*, September 2012.

[17] F. Damiani, J. Dovland, E. B. Johnsen, O. Owe, I. Schaefer, and I. C. Yu. A transformational proof system for delta-oriented programming. In *16th International Software Product Line Conference*, September 2012.

[18] F. de Boer, M. Helvensteijn, and J. Winter. A modal logic for abstract delta modeling. In *16th International Software Product Line Conference*, September 2012.

[19] D. Dhungana, T. Neumayer, P. Grunbacher, and R. Rabiser. Supporting evolution in model-based product line engineering. In *SPLC*, pages 319–328. IEEE Press, 2008.

[20] D. Dhungana, D. Seichter, G. Botterweck, R. Rabiser, P. Grünbacher, D. Benavides, and J. A. Galindo. Configuration of multi product lines by bridging heterogeneous variability modeling approaches. In E. Almeida, T. Kishi, C. Schwanninger, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 1, pages 120–129. IEEE Computer Society, August 2011.

[21] J. Dong. An ideal schema design of an industrialized PL-ISEE database platform. In *Advances in Intelligent Structure and Vibration Control, International Conference on Intelligent Structure and Vibration Control (ISVC 2012), Chongqing, China*, pages 341–345. Trans Tech Publications, March 2012.

[22] S. El-Sharkawy, S. Dederichs, and K. Schmid. From Feature Models to Decision Models and Back Again: An Analysis Based on Formal Transformations . In *16th International Software Product Line Conference*, 2012.

[23] J. A. Galindo, F. Roos-Frantz, J. Garcia-Galàn, and A. Ruiz-Cortés. Extracting orthogonal variability models from debian repositories. In I. Schaefer, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 2. IEEE Computer Society, August 2011.

[24] J. Garcia-Galàn, P. Trinidad, J. A. Galindo, and A. Ruiz-Cortés. Tool supported error detection and explanations on feature models. In I. Schaefer, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 2. IEEE Computer Society, August 2011.

[25] Y. Ghanam and F. Maurer. Extreme Product Line Engineering: Managing Variability and Traceability via Executable Specifications. In *AGILE*, pages 41–48. IEEE Press, 2010.

[26] S. Gnesi and M. Petrocchi. Towards an executable algebra for product lines. In *16th International Software Product Line Conference*, September 2012.

[27] W. Heider, R. Froschauer, P. Gruenbacher, R. Rabiser, and D. Dunghana. Simulating evolution in model-based product line engineering. *Information and Software Technology*, 52(7):758–769, 2010.

[28] A. Heuer, C. Budnik, S. Konrad, K. Lauenroth, and K. Pohl. Formal Definition of Syntax and Semantics for Documenting Variability in Activity Diagram. In *Software Product Line Conference*, volume 1, pages 62–76, 2010.

[29] M. Hinchey. Formally Specifying Families of Space Exploration Missions. In *Software Product Line Conference*, volume 2, page 73, 2010.

[30] M. Hinchey. Families (of products) in space. In I. Schaefer, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 1. IEEE Computer Society, August 2011.

[31] Y. Huilin and W. Zhang. Formal definition of feature models to support software product line evolution. In *International Conference on Software Engineering Research and Practice*, pages 349–355, 2008.

[32] IEC Standard 62304: Medical device software—Software life cycle processes. International Electrotechnical Commission, 2006.

[33] ISO Standard 25119: Tractors and machinery for agriculture and forestry—Safety-related parts of control systems. International Organisation for Standardization, 2010.

[34] ISO Standard 26262: Road vehicles—Functional safety. International Organisation for Standardization, 2009.

[35] L. Jing, J. Dehlinger, S. Hongyu, and R. Lutz. State-based modeling to support the evolution and maintenance of safety-critical software product lines. In *IEEE Symposium and Workshop on Engineering of Computer Based Systems*, pages 1–10. IEEE Press, 2007.

[36] S. Joerges, A.-L. Lamprecht, T. Margaria, I. Schaefer, and B. Steffen. A constraint-based variability modeling framework. *International Journal on Software Tools for Technology Transfer (STTT)*, 2012.

[37] J. A. Kim. Case Study of Software Product Line Engineering in Insurance Product. In *Software Product Line Conference*, page 512. Springer-Verlag, 2010.

[38] M. Kim, S. Park, V. Sugumaran, and H. Yang. Managing requirements conflicts in software product lines: A goal and scenario based approach. *Data & Knowledge Engineering*, 61(3):417–432, 2007.

[39] N. Kozuka and Y. Ishida. Building a product line architecture for variant-rich enterprise applications using a data-oriented approach. In I. Schaefer, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 2. IEEE Computer Society, August 2011.

[40] A. Lamprecht, T. Margaria, I. Schaefer, and B. Steffen. Comparing structure-oriented and behaviour-oriented variability modeling for workflows. In *1st International Workshop on Eternal Systems (EternalS'11), Communications in Computer and Information Science (CCIS)*. Springer, 2011.

[41] K. Lauenroth and K. Pohl. Dynamic consistency checking of domain requirements in product line engineering. In *RE*, pages 193–202. IEEE Press, 2009.

[42] H. Lochmann. *HybridMDSD: Multi-Domain Engineering with Model-Driven Software Development Using Ontological Foundations*. Fakultät Informatik, TU Dresden, Germany, 2010.

[43] P. S. M. Cordy, P. Heymans and A. Legay. Behavioural Modelling and Verification of Real-Time Software Product Lines. In *16th International Software Product Line Conference*, 2012.

[44] A. Malaer and M. Lampe. SimPL—a simple software production line for end user development. In *APSEC*, pages 179–186, 2008.

[45] E. Martínez and G. Schneider. Automated Analysis of Conflicts in Software Product Lines. In *Software Product Line Conference*, volume 2, pages 75–82, 2010.

[46] D. Mellado, E. Medina, and M. Piattini. Security requirements variability for software product lines. In *ARES*, pages 1413–1420. IEEE Press, 2008.

[47] E. Murugesupillai, B. Mohabatti, and D. Gasevic. A preliminary mapping study of approaches bridging software product lines and service-oriented architectures. In I. Schaefer, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 2. IEEE Computer Society, August 2011.

[48] R. Muschevici, D. Clarke, and J. Proença. Feature petri nets. In G. Botterweck, S. Jarzabek, T. Kishi, J. Lee, and S. Livengood, editors, *14th International Software Product Line Conference Proceedings*, volume 2, pages 99–106, U.K., September 2010. Lancaster University.

[49] M. Naeem and R. Heckel. Towards matching of service feature models based on linear logic. In I. Schaefer, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 2. IEEE Computer Society, August 2011.

[50] S. Nakajima. Non-clausal Encoding of Feature Diagram for Automated Diagnosis. In *Software Product Line Conference*, volume 1, pages 420–424, 2010.

[51] A. S. Nascimento, C. M. Fischer Rubira, and J. Lee. An SPL approach for adaptive fault tolerance in SOA. In I. Schaefer, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 2. IEEE Computer Society, August 2011.

[52] A. Noehrer and A. Egyed. Optimizing user guidance during decision-making. In E. Almeida, T. Kishi, C. Schwanninger, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 1, pages 25–34. IEEE Computer Society, August 2011.

[53] Y. Oh, D. H. Lee, S. Kang, and J. H. Lee. Extended Architecture Analysis Description Language for Software Product Line Approach in Embedded Systems. In *International Conference on Formal Methods and Models for Codesign*, pages 87–88. IEEE Press, 2007.

[54] Oki Electric Industry Co. Ltd.: Financial and strategic analysis review. Global Markets Direct, December 2009.

[55] S. Oster, F. Markert, and P. Ritter. Automated Incremental Pairwise Testing of Software Product Lines. In *Software Product Line Conference*, volume 1, pages 196–210, 2010.

[56] Software development in the product lifecycle. Ovum, April 2010. 32 pages.

[57] K. Pohl, G. Böckle, and F. Van Der Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques*. Springer, 2005.

[58] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: the next computing revolution. In *47th Design Automation Conference*, pages 731–736, 2010.

[59] H. B. G. Ribeiro, E. Santana de Almeida, and S. R. d. Lemos Meira. An approach for implementing core assets in service-oriented product lines. In I. Schaefer, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 2. IEEE Computer Society, August 2011.

[60] S. T. Ruehl and U. Andelfinger. Applying software product lines to create customizable software-as-a-service applications. In I. Schaefer, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 2. IEEE Computer Society, August 2011.

[61] H. Sabouri and R. Khosravi. An Effective Approach for Verifying Product Lines in Presence of Variability Models. In *Software Product Line Conference*, volume 2, pages 113–120, 2010.

[62] I. Schaefer, L. Bettini, V. Bono, F. Damiani, and N. Tanzarella. Delta-oriented programming of software product lines. In *Proc. of 15th Software Product Line Conference (SPLC 2010)*, September 2010.

[63] I. Schaefer and F. Damiani. Pure Delta-oriented Programming. In S. Apel, D. Batory, K. Czarnecki, F. Heidenreich, C. Kästner, and O. Nierstrasz, editors, *Proc. 2nd International Workshop on Feature-Oriented Software Development (FOSD'10) Eindhoven, The Netherlands*, pages 49–56. ACM Press, 2010.

[64] I. Schaefer and R. Hähnle. Formal methods in software product line engineering. *IEEE Computer*, 44(2):82–85, February 2011.

[65] I. Schaefer, R. Rabiser, D. Clarke, and et al. Software diversity: State of the art and perspectives. *International Journal on Software Tools for Technology Transfer (STTT)*, 2012.

[66] K. Schmid and A. Rummler. Cloud-based software product lines. In *16th International Software Product Line Conference*, September 2012.

[67] J. Schroeter, P. Mucha, M. Muth, K. Jugel, and M. Lochau. Dynamic configuration management of cloud-based applications. In *16th International Software Product Line Conference*, September 2012.

[68] H. Serajzadeh and F. Shams. The application of swarm intelligence in service-oriented product lines. In I. Schaefer, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 2. IEEE Computer Society, August 2011.

[69] C. Snook, M. Poppleton, and I. Johnson. Rigorous engineering of product-line requirements: a case study in failure management. *JIST*, 50(1–2):112–129, 2008.

[70] R. Tahwid and D. C. Petriu. Automatic derivation of a product performance model from a software product line model. In E. Almeida, T. Kishi, C. Schwanninger, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 1, pages 80–89. IEEE Computer Society, August 2011.

[71] O. Takizawa, F. Akaishii, and T. Hayase. Software-platform construction technology. *Toshiba Leading Innovation*, 64(4):36–39, 2009. In Japanese.

[72] T. Thuem, C. Kästner, S. Erdweg, and N. Siegmund. Abstract features in feature modeling. In E. Almeida, T. Kishi, C. Schwanninger, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 1, pages 191–202. IEEE Computer Society, August 2011.

[73] Q. M. Tian, X. Y. Chen, L. Jin, P. Pan, and C. Ying. Asset-based requirement analysis in telecom service delivery platform domain. In *NOMS*, pages 815–818. IEEE Press, 2008.

[74] N. Ubayashi, S. Nakajima, and M. Hirayama. Context-Dependent Product Line Practice for Constructing Reliable Embedded Systems. In *Software Product Line Conference*, volume 1, pages 1–15, 2010.

[75] T. Vale, E. Almeida, S. Meira, and G. Figueiredo. A study on service identification methods for software product lines. In *16th International Software Product Line Conference*, September 2012.

[76] P. van den Broek. Intersection of Feature Models. In *16th International Software Product Line Conference*, September 2012.

[77] F. J. van der Linden, K. Schmid, and E. Rommes. *Software product lines in action: the best industrial practice in product line engineering.* Springer-Verlag, 2007.

[78] L. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner. A Break in the Clouds: Towards a Cloud Definition. *ACM-SIGCOMM*, 39(1):50–55, 2009.

[79] K. Villela, T. Arif, and D. Zanardini. Towards product configuration taking into account quality concerns. In *16th International Software Product Line Conference*, September 2012.

[80] M. Voelter and E. Visser. Product line engineering using domain-specific languages. In E. Almeida, T. Kishi, C. Schwanninger, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 1, pages 70–79. IEEE Computer Society, August 2011.

[81] K. Wnuk, B. Regnell, and L. Karlsson. Feature Transition Charts for Visualization of Cross-Project Scope Evolution in Large-Scale Requirements Engineering for Product Lines. In *Workshop on Requirements Engineering Visualization*, pages 11–20, 2009.

[82] P. Wong, N. Diakov, and I. Schaefer. Modelling software product lines using HATS approach – a Fredhopper case study. In I. Schaefer, I. John, and K. Schmid, editors, *15th International Software Product Line Conference Proceedings*, volume 2. IEEE Computer Society, August 2011.

Glossary

Terms and Abbreviations

ABS Abstract Behavioral Specification language. An executable class-based, concurrent, object-oriented modeling language based on Creol, created for the HATS project.

Application engineering Application engineering is a process that builds a single product by reusing artifacts from a product line artifact base.

BDD Binary Decision Diagram.

Creol A type-safe object-oriented model for distributed concurrent systems. Creol targets distributed objects by a looser coupling of method calls and synchronization.

CSP Constraint Satisfaction Problem.

EU-IST European Union—Information Society Technologies.

Family engineering Family engineering is a process that builds reusable artifacts that are stored in a product line artifact base. See also product line artifact base.

GPL The GNU General Public License (GNU GPL or simply GPL) is the most widely used free software license. It is a copyleft license for general use, which means that derived works can only be distributed under the same license terms.

ICT Information and Communication Technologies.

IT Information Technology.

LGPL The GNU Lesser General Public License (formerly the GNU Library General Public License) is a free software license that was designed as a compromise between the strong-copyleft GNU General Public License and permissive licenses. The LGPL places copyleft restrictions on the program itself but does not apply these restrictions to other software that merely links with the program. The LGPL is primarily used for software libraries, although it is also used by some stand-alone applications.

Product line artifact base A repository in a software product line containing all reusable artifacts.

SAT Boolean Satisfiability Problem.

SME Small and Medium Enterprises.

SMT Satisfiability Modulo Theories problem is a decision problem for logical formulas with respect to combinations of background theories expressed in classical first-order logic with equality.SMT can be thought of as a form of the constraint satisfaction problem and thus a certain formalized approach to constraint programming.

SPL engineering A development methodology for software product families. It splits development into family engineering and application engineering processes. See also family engineering and application engineering.

Software product family A family of software systems with well-defined commonalities and variabilities.

SWOT analysis It is a strategic planning method used to evaluate the Strengths, Weaknesses/Limitations, Opportunities, and Threats involved in a project or in a business venture.

VDM++ extension of the specification language of the Vienna Development Method (VDM) to support the modeling of object-oriented and concurrent systems.