HATS Project Overview and Introduction

Reiner Hähnle

Technische Universität Darmstadt

HATS Annual Review Meeting 2012

HATS
Highly Adaptable and Trustworthy Software using Formal Models

SEVENTH FRAMEWORK PROGRAMME

http://www.hats-project.eu

HATS Facts

HATS: Highly Adaptable & Trustworthy Software Using Formal Models

- FP7 FET focused call Forever Yours
- Project started 1 March 2009, 48 months runtime
- Integrated Project, academically driven
- 10 academic partners, 2 industrial research, 1 SME
  - Coordinator moved to TU Darmstadt on 1 Sep, 2012
  - New beneficiary TUD
- 8 countries
- 805 PM, EC contribution 5,64 M€ over 48 months
- Associated with FP7 Coordination Action: EternalS
  - Trustworthy Eternal Systems via
    Evolving Software, Data and Knowledge

HATS Consortium & Lead Researchers

The highlighted people are present

Hähnle, Bubel	Technische Univ. Darmstadt (Sci. Coor.)	DE
Ahrendt, Waborg	Chalmers Tekniska Högskola (Adm. Coor.)	SE
Johnsen, Schlatte	Universitetet i Oslo	NO
Dam, Gurov, Meinke	Kungliga Tekniska Högskolan	SE
Albert, Barthe, Puebla	Universidad Politécnica de Madrid/IMDEA	ES
Poetzsch-Heffter	University of Kaiserslautern	DE
Sangiorgi, Bravetti	Università di Bologna	IT
De Boer	Centrum voor Wiskunde en Informatica	NL
Hagalisletto, Østvold	Norsk Regnesentral	NO
Diakov, Wong	Fredhopper	NL
Villela, Arif	Fraunhofer IESE	DE
Clarke, Proença	Katholieke Universiteit Leuven	BE
Uustalu, Laud	Institute of Cybernetics	EE

What Does HATS?

In a nutshell, we . . .

develop a tool-supported formal modeling language (ABS)

for the design, analysis, and implementation of

highly adaptable software systems characterized by a

high expectations on trustworthiness

for target software systems that are . . .

- concurrent, distributed
- object-oriented
- built from components
- adaptable (variability, evolvability), hence reusable

Main focus: Software Product Line Engineering

Motivation

Why formal?

- informal notations can't describe software behavior with rigor: concurrency, modularity, correctness, security, resources . . .
- formalization $\Rightarrow$ more advanced tools
  - more complex products
  - higher automation: cost-efficiency

Motivation

Why formal?

- informal notations can't describe software behavior with rigor: concurrency, modularity, correctness, security, resources ...
- formalization ⇒ more advanced tools
  - more complex products
  - higher automation: cost-efficiency

Why adaptable?

- software rich in features and variants, many deployment scenarios
- changing requirements (rapid technological/market pace)
- evolution of software in unanticipated directions
- language-supported adaptability is a key to successful reuse

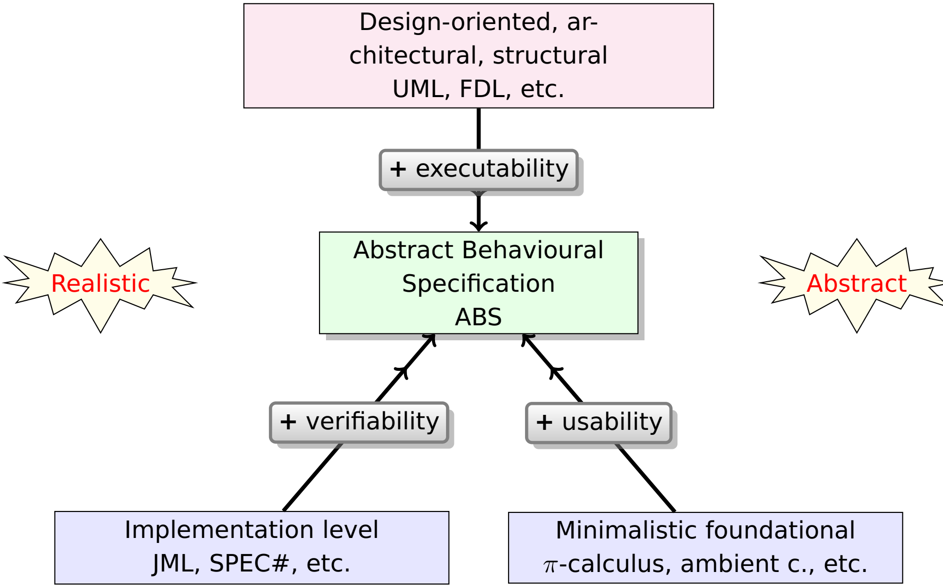
Mind the Gap!

Design-oriented, architectural, structural
UML, FDL, etc.

Implementation level
JML, SPEC#, etc.

Minimalistic foundational
π-calculus, ambient c., etc.

Mind the Gap!


Design-oriented, architectural, structural
UML, FDL, etc.
+ executability
Realistic
Abstract Behavioural Specification
ABS
Abstract
+ verifiability
+ usability
Implementation level
JML, SPEC#, etc.
Minimalistic foundational
π-calculus, ambient c., etc.

How?

A tool-supported formal method for
building highly adaptable and trustworthy software

How?

A tool-supported formal method for
building highly adaptable and trustworthy software

Main ingredients

1. Executable, formal modeling language for adaptable software: Abstract Behavioral Specification (ABS) language
2. Tool suite for ABS/executable code analysis & development:

   Analytic functional/behavioral verification, resource analysis, feature consistency, RAC, types, TCG, visualization

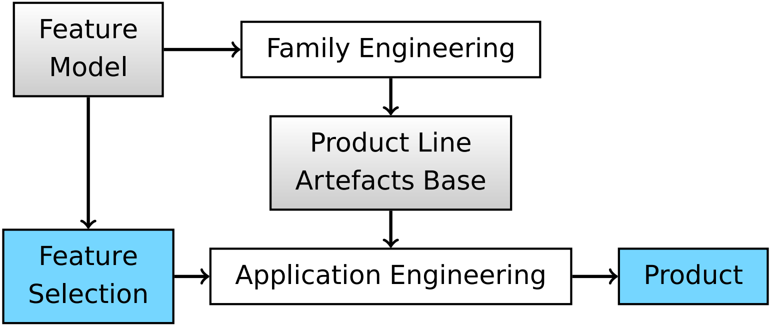
   Generative code generation, model mining, monitor inlining, . . .

   Develop methods in tandem with ABS to ensure feasibility
3. Methodological and technological framework integrating HATS tool architecture and ABS language

Important Project Principles (I)

Ensuring relevance

- Apply to empirically highly successful development method: Software product line engineering (PLE)
- Thorough requirements analysis, continuous evaluation



Important Project Principles (II)

Feasibility: ensure that analysis methods scale up

Develop analysis methods in tandem with ABS language

Incrementality

- Delta modeling, delta specification, delta verification

Compositionality

- Concurrency model
- Proof systems

Automation

- Type systems for Near/Far-analysis, deadlocks, safe products
- Resource analysis, test case generation

Ease of Usage

- Integrate into design methodology/workflow
- Integrate into standard IDE (Eclipse)

Important Project Principles (III)

Early evaluation

- Develop Core ABS first

Local Contracts, Assertions
Syntactic Modules
Asynchronous Communication
Concurrent Object Groups
Imperative Language
Object Model
Pure Functional Programs
Algebraic Data Types

Important Project Principles (III)

Early evaluation

- Develop Core ABS first
- Layered language design

Delta Modeling Languages: μTVL, DML, CL, PSL

Component Model

Deployment Components: Real-Time ABS

Local Contracts, Assertions

Syntactic Modules

Asynchronous Communication

Concurrent Object Groups

Imperative Language

Object Model

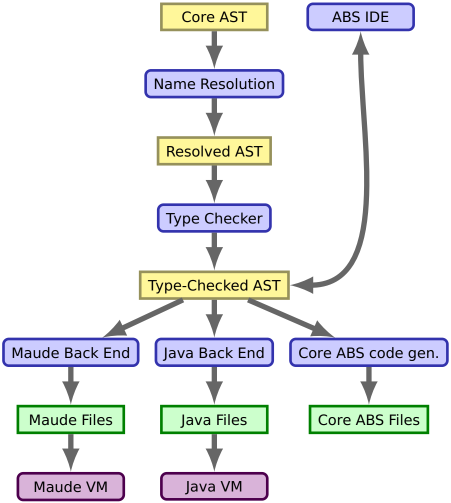
Pure Functional Programs

Algebraic Data Types

Important Project Principles (III)

Early evaluation

- Develop Core ABS first
- Layered language design
- Provide tools early



The Main Innovations of HATS

ABS—a formal, executable, abstract, behavioral modeling language

- Cutting-edge research on modeling of concurrent, OO systems
- Combines state-of-art in verification, concurrency, specification, and programming languages communities
- Tailored to model feature-rich and evolvable systems

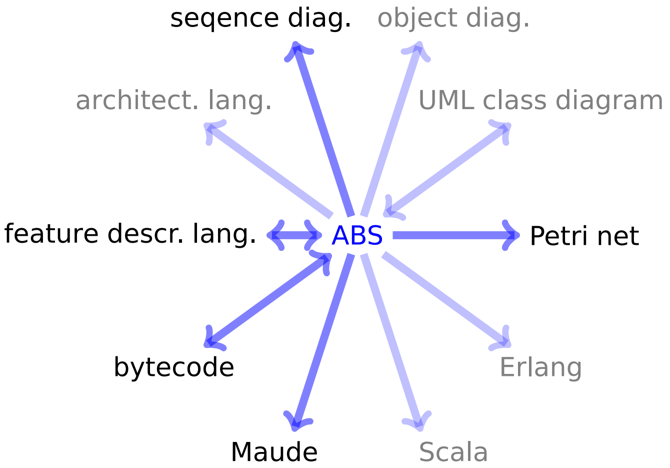
Scalable technologies developed in tandem with ABS

- Incremental, compositional
- Analytic as well as generative technologies

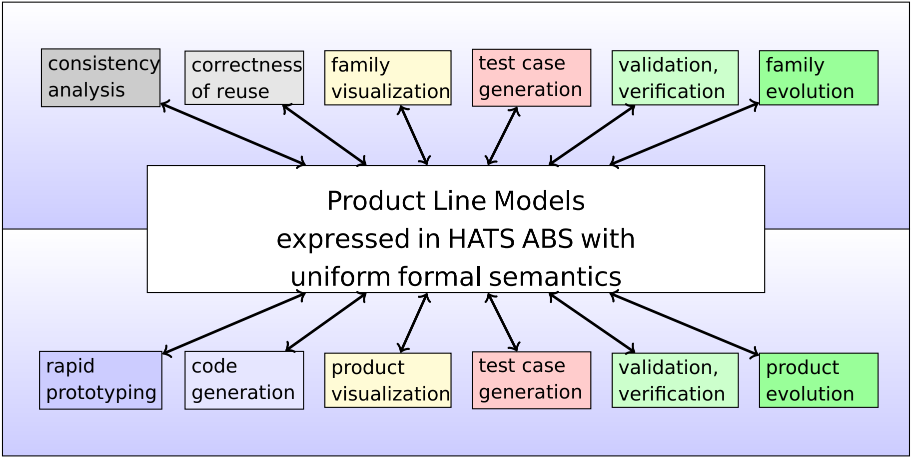
Formalization of PLE-based development as main application

- Leveraging formal methods tools to Product Line Engineering
- Define FM-based development methodology for PLE

Vision: A Single-Source Technology for Highly Adaptive, Concurrent Software Systems



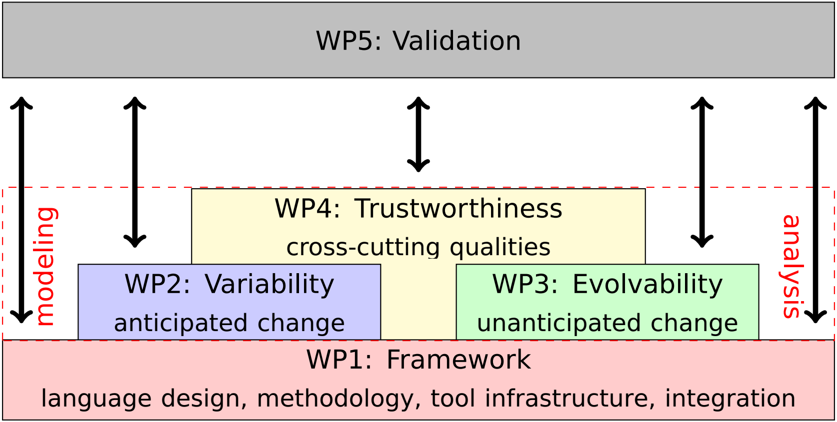
Family Engineering



Application Engineering

[Schaefer & Hähnle, IEEE Computer, Feb. 2011]

Work Organisation in HATS



Third Year Deliverables

Del. no.	Deliverable name	WP no.	Lead beneficiary	Nature	Dissemination level	Delivered	Actual/ Forecast	Comments
D2.1	Configuration Deployment	2	UIO	report	public	✔	35/35	shifted from PM30
D2.3	Testing, Debugging and Visualization	2	TUD	report	public	✔	36/36	
D2.4	Types for Variability	2	BOL	report	public	✔	36/36	
D3.2	Model mining	3	NR	report	public	✔	36/36	
D4.1	Security	4	IMDEA	report	public	✔	36/36	
D5.3	Evaluation of modeling	5	CWI	report	public	✔	36/36	validated M2

Milestones

Milestone no.	Milestone name	WP no.	Lead bene-ficiary	Achieved	Actual/ Forecast	Comments
M1	Core language and methodology	1	UIO	✔	12/12	D1.1a, D1.1b Validated in D5.2
M2	Modeling capabilities	1,2,3	UKL	✔	24/24	D1.2, D2.2b, D3.1b Validated in D5.3
M3	Analysis and development methods	2,3,4	TUD	✔	36/36	D2.1, D2.3, D2.4, D2.5, D3.2, D3.4, D4.1, D4.2 To be validated in D5.4

Active Technical Work Tasks in Third Year

Nr	Name	Lead	Start	Deliv.	Comment
1.3	Analysis	TUD	PM25	—	Early start Milestone M4 formerly CTH
1.4	System derivation and code generation	IoC	PM25	—	Milestone M4
1.5	Integrated tool platform	NR	PM25	—	
2.1	A configurable deployment architecture	UIO	PM7	2.1	Extended Milestone M3 Finished
2.3	Testing, debugging, and visualization	TUD	PM19	2.3	Milestone M3 Finished formerly CTH
2.4	Types for variability	BOL	PM13	2.4	Milestone M3 Finished
2.6	Refinement and abstraction	UKL	PM25	—	Milestone M4

Active Technical Work Tasks in Third Year Cont'd

Nr	Name	Lead	Start	Deliv.	Comment
3.2	Model mining	NR	PM19	3.2	Milestone M3 Finished
3.3	Hybrid analysis for evolvability	KUL	PM25	—	
3.5	Autonomously evolving systems	KTH	PM25	—	Early start
4.1	Security	IMDEA	PM13	4.1	Milestone M3 Finished
4.3	Correctness	BOL	PM25	—	
4.4	Auto configuration and quality variability	FRG	PM31	—	Milestone M4
5.3	Evaluation of modeling	CWI	PM19	5.3	Milestone M2 Finished
5.4	Evaluation of tool and techniques	FRH	PM31	—	Milestone M4

15 work tasks finished, 9 continue, all tasks started

Third Year Objectives & Results: Milestone 3

Development Methods

- Delta modeling workflow based on abstract delta modeling (T5.3)

Analysis Methods in particular, at feature modeling level

- Near/far location type analysis (T1.3), deadlock analysis (T4.3)
- Type system for checking type-safety of delta models (T2.4)
- Glass box test case generator and ABSUnit framework (T2.3)
- Model checker for products of a SW product line (T1.3)

Generative Methods

- Model mining from code, traces, product descriptions (T3.2)
- Automatic construction of crypto protocol implementation (T4.1)
- Scala backend for ABS (T1.4)

Security Policies

- Information-flow type system for core ABS (T4.1)
- Logic-based and dynamic enforcement of security policies (T4.1)

Previous year: behavioral verification (T2.5), resource analysis (T4.2)

Further Third Year Objectives & Results

Dynamic Aspects, Evolvability

- Deployment components to model low-level notions (T2.1)
  - Schedulers, load, bandwidth
  - Real-time ABS
- Abstract failure model and type system (T2.1, T2.4)
- ABS component model (T2.1, T3.3)
- ABS-NET: semantics for network-aware runtime ABS (T3.5)

Evaluate Modeling

- Fredhopper Access Server replication system (T5.3)
  - Modeled with Full ABS as basis for validation of M2
- Some models in S.P.L.O.T. feature model repository (T5.3)

Implementation of Recommendations from Reviewers

Implementation of Recommendations from Reviewers

1. The ABS language occupies a [. . .] “niche” by focussing on an intermediate level between abstract models and code. The project is encouraged [. . .], but it should take care that the links to the more “classical” levels of abstraction are well studied . . .
   - Task 2.6 Refinement and Abstraction:
     Correct refinements to/from ABS artefacts, including trace semantics, timed automata, and rule-based compilation
   - Task 3.2 Model Mining:
     Relate ABS to automata-based abstractions and to the Java Message Service
   - Task 3.3 Evolvability for Hybrid Systems:
     A suitable notion of runtime component for ABS

Implementation of Recommendations from Reviewers

1. The ABS language occupies a [. . .] “niche” by focussing on an intermediate level between abstract models and code. The project is encouraged [. . .], but it should take care that the links to the more “classical” levels of abstraction are well studied . . .
   - Task 2.6 Refinement and Abstraction:
     Correct refinements to/from ABS artefacts, including trace semantics, timed automata, and rule-based compilation
   - Task 3.2 Model Mining:
     Relate ABS to automata-based abstractions and to the Java Message Service
   - Task 3.3 Evolvability for Hybrid Systems:
     A suitable notion of runtime component for ABS
2. Continue/improve successful development of the HATS tool suite
   - Task 1.5 Integrated Tool Platform:
     See demo in next session (task started 6 months earlier)

Implementation of Recommendations from Reviewers

3. In addition to the classification of the verification properties, the roles and relevance of these properties for product line engineering should be evaluated and a systematic approach for the use of verification properties should be developed
   - Task 2.4 identified type-safety of products as central
   - Extended to dynamic code updates in Task 3.3
   - Task 4.3 looks systematically at correctness properties
   - Evaluation of analysis tools: Task/Deliverable 5.4 PM 48
4. Clarify the advantages/drawbacks of Delta programming [. . .]; because of the decision for Delta modelling, the project should focus the work of the remaining tasks of WP2 on this technique
   - Delta modeling was concretized into Delta Modeling Workflow process, defined and evaluated in Deliverable D5.3
   - Evaluation of delta modeling to be continued in Task 5.4
   - Delta modeling part of full ABS, all subsequent tasks use it

Implementation of Recommendations from Reviewers

5. Continue to take integration seriously. Make an effort to increase joint publications which involve different project partners.
   - Of 46 publications in 3rd project year, no less than 13 (ca. 30%) involved more than one project partner
6. The exploitation plan is well under way; however, the sustainability of tools beyond the end of the project deserves more attention.
   - Some new EC/national project proposals based on ABS underway
   - TUD, UIO and UKL committed to maintain ABS tool chain beyond HATS
7. The new Estonian partner should be effectively involved into the project work and present its results at the next review.
   - In 3rd period IoC participated actively in Tasks 1.4 (lead), Task 2.4, Task 4.1, and Task 4.3
   - Co-authorship of D4.1
   - Today, progress report of T1.4, mentioned in T4.1 and T4.3
   - Organization of this review meeting, 5 publications

Implementation of Recommendations from Reviewers

8. Although not required by the DoW, we suggest studying the extension of the COSTA framework to adaptable systems [. . .]
   - Will be addressed in ongoing Task 4.4 Auto Configuration and Quality Variability, where UPM plans to develop incremental resource analysis in COSTABS
9. [. . .] The reviewers recommend [. . .] to take appropriate action for ensuring a continuous quality of the research in the areas where Ina Schaefer was leading the work.
   - Ina continues to contribute towards the project results:
     - Co-authorship of Deliverables D2.4 and D3.2
     - Five joint publications within HATS during 3rd period
     - Tenured post (TU Braunschweig) basis for continued collaboration
   - Ina's expertise successfully transferred to TUD, BOL, KUL, KTH, UIO
10. Although the quality of the deliverables was considerably better than in year 1, we recommend the following improvements: . . .
    - All recommendations have been implemented

Detailed Presentation of Activities and Results

Work Package	WP Leader	Presenters (Task Leaders)
WP1 Framework, Demo	UKL Poetzsch-Heffter	Østvold (T1.5) Schlatte (demo, T1.5) Bubel (T1.3) Uustalu (T1.4)
WP2 Variability	UIO Johnsen	Johnsen (T2.1) Bubel (T2.3) Sangiorgi (T2.4) Poetzsch-Heffter (T2.6)
WP3 Evolvability	KTH Dam	Østvold (T3.2) Proença (T3.3) Dam (T3.5)
WP4 Trustworthiness	UPM/IMDEA Albert	Barthe (T4.1) Bravetti (T4.3) Albert (T4.4)

Detailed Presentation of Activities and Results Cont'd

Work Package	WP Leader	Presenters (Task Leaders)
WP5 Validation	FRH Wong	de Boer (T5.3) Wong (demo, T5.4)
WP6 Dissemination and Training	FRG (Villela)	Arif (T6.1, T6.2) Sangiorgi (T6.3)
WP7 Management	CTH (admin) TUD (scientific) Waborg, Hähnle	Hähnle (T7.1) Waborg (T7.2)