Embedding Formal Methods into Systems Engineering

Helmut Veith
FORSYTE Group, TU Darmstadt
School of Computer Science, Carnegie Mellon University

Experience from Collaborations with Automotive / Avionic Industry
… a clear separation of concerns emerges: we might call them the mathematical concerns about correctness and the engineering concerns about efficiency.

... the discovery how to separate them rigorously in our thinking is relatively young, and even when aware of this possibility, we often fall back into our old bad habits.
The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise. [Turing Award Lecture]
The purpose of abstraction is \textit{not} to be vague, but to create a new semantic level in which one can be absolutely precise. [Turing Award Lecture]
The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise. [Turing Award Lecture]
Abstraction

The purpose of abstraction is *not* to be vague, but to create a new semantic level in which one can be absolutely precise. [Turing Award Lecture]
The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise. [Turing Award Lecture]
The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise. [Turing Award Lecture]
The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise. [Turing Award Lecture]

The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise. [Turing Award Lecture]
The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise. [Turing Award Lecture]

… the only problems we can really solve in a satisfactory manner are those that finally admit a nicely factored solution.

Feature Driven Development
- functional requirements
- top down / bottom up implementation
- code topology reflects features

Divide and Conquer, Encapsulation, Components …
Compilers: A Microcosm of Model Based Design

high level program

\[\downarrow\]

executable

\[\downarrow\]

abstract model

... with mathematical precision.

concretization

Compiler
... analyzes (one) input program
... recognizes certain program errors
... resolves language nondeterminism
... knows target hardware
... performs optimizations
... is trusted to be correct

Formal Methods
- Model Analysis / Validation
- Model Transformation
- Model Optimization

constraints

... -O1 -O2 -O3 -Os

... with mathematical precision.

solution

Compiler
... analyzes (one) input program
... recognizes certain program errors
... resolves language nondeterminism
... knows target hardware
... performs optimizations
... is trusted to be correct

Formal Methods
- Model Analysis / Validation
- Model Transformation
- Model Optimization
Formal Methods

Computer-aided mathematically precise reasoning methodologies about systems (and) models.

SAT and SMT Solving, Model Checking, Synthesis, Testing, Theorem Proving, …

A. M. Turing Award 2007
for Model Checking

Formulas have always frightened me.

At the same time, I have a warm appreciation for well-designed formalisms that enable us to do things that I couldn’t possibly do without them.

[My Hopes of Computing Science]

Model Checking

critical property
model checking
program
compilation
executable

code assertions
absence of deadlocks
termination
correct API use
path feasibility
memory violations
safety & liveness …

Property violations documented by program traces! I know a bug when I see it.
Model Checking

critical property
  ↓ model checking
program
  ↓ compilation
executable

Property violations documented by program traces! I know a bug when I see it.

H. Veith 5/2008

Formal Methods Ctd.

compiler          correctly typed?
SAT / SMT decision proc. test case feasible?
deployment possible?
testing runtime errors? coverage?

theorem proving (ISABEL, ACL2, PVS)
"requires specialized expertise that precludes its broad use in industry"
(Bob Kurshan, Cadence)

H. Veith 5/2008
Software Model Checking

- 2000s: development of industrial strength C model checkers
- “… rivals theorem proving for many verification tasks” (Rushby)
- Microsoft product for Windows device driver verification

Software Engineering

(i) separation of correctness and efficiency
(ii) abstraction and hierarchy
Systems Engineering

Multidisciplinary Effort
- Mechanical Engineering
- Electrical Engineering
- Computer Science

Added value in automobile manufacturing

70 ECUs Inside

Power consumption?
Safety?
Certification?
ECU distribution?
Cost per car?
HW or SW?
FPGA or ASIC?
Multicore?

Weight?
Repair logistics?
Component Reuse?
Intellectual Property?
Product Lines?
Variant Management?
Physical Environment?
Obsolescence?
Dependability?

Mechanical engineering process?
ECU distribution?
Deployment strategy?
Certification?
Power consumption?
Execution Time?
Memory use?
Safety?
Cost per car?

Computer-aided Car Compilation?

H. Veith 5/2008
System Design Challenge

Constraints
- Software: algorithms, protocols, architectures, compilers, …
- Hardware: speed, memory, power, failure rates, weight, size…
- Environment: performance, robustness, …

integration / solution / compilation

SYSTEM BLUEPRINT

H. Veith 5/2008
System Design Challenge

Constraints

- Software: algorithms, protocols, architectures, compilers, …
- Hardware: speed, memory, power, failure rates, weight, size…
- Environment: performance, robustness, …
- Cost: development, HW, safety, certification, warranty, testing, …
- Time: time to market, product lines, variants, component reuse …

integration / solution / compilation

SYSTEM BLUEPRINT

informatics + physics = emerging theory

Embedded Systems Design Challenge

business + law + culture

automotive, avionic, …

H. Veith 5/2008
System Design Challenge

Constraints

• Software: algorithms, protocols, architectures, compilers, …
• Hardware: speed, memory, power, failure rates, weight, size…
• Environment: performance, robustness, …
• Cost: development, HW, safety, certification, warranty, testing, …
• Time: time to market, product lines, variants, component reuse …

integration / solution / compilation

SYSTEM BLUEPRINT
System Design Challenge

Constraints
- heterogeneous
- global / horizontal
- analytical + discrete
- transcend design layers
- multiple optimization criteria

integration / solution / compilation

SYSTEM BLUEPRINT

Guru Principle
- knows trade-offs
- holds global view
- maintains IP

H. Veith 5/2008
System Design Challenge

Constraints:
- heterogeneous
- global / horizontal
- analytical + discrete
- transcend design layers
- multiple optimization criteria

Integration / Solution / Compilation

SYSTEM BLUEPRINT

Guru Principle:
- Multiple models & disciplines
- Transition between models
- Model validation and integration
- Multiple orthogonal views
- Exceeding complexity

H. Veith 5/2008
System Design Challenge

Constraints
- heterogeneous
- global / horizontal
- analytical + discrete
- transcend design layers
- multiple optimization criteria

integration / solution / compilation

SYSTEM BLUEPRINT

Multiple models & disciplines
Transition between models
Model validation and integration
Multiple orthogonal views
Exceeding complexity

H. Veith 5/2008
System Design Challenge

Constraints
heterogeneous
global / horizontal
analytical + discrete
transcend design layers
multiple optimization criteria

Integration / solution / compilation

SYSTEM BLUEPRINT

Multiple models & disciplines
Transition between models
Model validation and integration

Guru Principle
Multiple models & disciplines
Transition between models
Model validation and integration

SYSTEM BLUEPRINT

Multiple orthogonal views
Exceeding complexity
System Design Challenge

Constraints
heterogeneous
global / horizontal
analytical + discrete
transcend design layers
multiple optimization criteria

integration / solution / compilation

SYSTEM BLUEPRINT

Multiple models & disciplines
Transition between models
Model validation and integration

Guru Principle
Multiple models & disciplines
Transition between models
Model validation and integration

Multiple orthogonal views
Exceeding complexity

Exceeding complexity

H. Veith 5/2008

SYSTEM BLUEPRINT
System Design Challenge

**Constraints**
- heterogeneous
- global / horizontal
- analytical + discrete
- transcend design layers
- multiple optimization criteria

**Guru Principle**
- Multiple models & disciplines
- Transition between models
- Model validation and integration
- Multiple orthogonal views
- Exceeding complexity
- Formal methods needed

**SYSTEM BLUEPRINT**

H. Veith 5/2008
System Design Challenge

Constraints
- heterogeneous
- global / horizontal
- analytical + discrete
- transcend design layers
- multiple optimization criteria

"The clever programmer is fully aware of the strictly limited size of his own skull; therefore, he approaches the programming task in full humility, and among other things he avoids clever tricks like the plague." - H. Veith 5/2008

Case Study

Automotive Industry

- Mechanical engineering culture
- Software recognized as strategic IP
- No software engineering process
- Jungle of models and ontologies
- Excessive warranty costs necessitate change

Project Goals
- Domain exploration and assessment
- Seamless modeling framework
- Prototype automotive tool chain
- Accompanying software engineering process
- Technology proof of concept
Seamless Modeling

Component View
Architectures

Functional View
Function Hierarchy

Requirements
Model Integration

Functional Hierarchy

Logical Architecture

Behavior

Test Vehicle

Laboratory Vehicle

Component Test

Model Integration

System Integration

time

Automotive Tool Chain

COLA model:
Model-level verification, testing, simulation

Clumped COLA model

Generated C code for each cluster

Execution time estimation

H. Veith 5/2008
COLA Component Language

Refinement
- Automaton
- Network
- Block
- Legacy

Properties
- SALT
- Hardware properties
- Requirements
- Deployment constraints
- Execution time estimations

H. Veith 5/2008
Automotive Tool Chain

COLA model:
Model-level verification, testing, simulation

Clustered COLA model

Generated C code for each cluster

Execution time estimation

Relating Efficiency and Computation

Time = Power = Hardware = Money

► Architecture
► Understanding execution time
Case Study
Avionic Software

Deployment decisions trial and error

Project Goals
- predictability and efficiency
- simplified automated deployment
- resource aware composition
- reusability of function units (IMA - AADL)
- execution time estimation

Avionic Software Analysis

Analysis and modeling of the hardware and software

Hardware analysis

Hardware profile ("HW Data Sheet")
HW-parameters for execution time

Symbolic execution time estimation (SW -> HW)

Software analysis

Software profile ("SW Data Sheet")
SW-parameters for execution time
Avionic Software Analysis

Hardware profile → Software profile → Model representation

Analysis/Validation → Refinement → Furness

SYSTEM BLUEPRINT

Foundational Research

Formal Timing Analysis Suite for Real Time Programs
Advanced white box test case generation powered by model checking for time measurement

(jointly with Real Time Systems Group, TU Vienna)

model checker = query engine → program = database

Automatic Extraction of Loop Bounds from C programs

… software engineering requires (cruelly) hard science for its support.

H. Veith 5/2008
Dijkstra Revisited

Challenges

Challenge 1
Multi-View Modeling Framework

Challenge 2
Heterogeneous System Composition

(i) separation of correctness and efficiency
(ii) abstraction and hierarchy

Challenge 3: Scale !!!

Challenge 4: Tools !!!

Can the entertainment unit affect the engine controller?
What is the power consumption of class X?
Can we implement this function in an ASIC?
Can we deploy the entertainment system on this car?
How many car variants needed for test driving?
Can you run integration tests for this architecture?
Does your geographical database meet hard deadlines?
Thank you for your attention!