# Models, Abstractions and Architectures in Component-Based Engineering

***or***

## Musings on better ways to specify complex systems

Steve Vestal
Honeywell Labs
Steve.Vestal@Honeywell.com

Honeywell

7 July 2005

# Goals

Honeywell

How do we do a better job of specifying complex things?

By having a theory of complex specifications.

In theory, theory and practice are the same, but in practice they are not.

Specifications are always incomplete, they focus on what is

- distinct from other products
- distinct from common practice
- critical (not left to implementer's whim)
- nonobvious (not left to implementer's capabilities)

Specifications always have non-formalizable elements.

# What is a Specification?

Honeywell

## Classical

Requirements specification defines "what"
Design specification defines "how"

## Recent

Domain specification defines problem & operational context
Functional specification defines "what"
Design specification defines "how"

## Practice

Great variability in number, level of detail and abstraction
Influenced by market segmentation, technology & legacies

# Deliberate Obtuseness

Honeywell

**One person's design decision is another person's derived requirement.**

**I won't draw distinctions between specifications for e.g.**

- Domain models
- Requirements
- Function
- Design
- Interface
- Implementation

**One person's system is another person's component.**

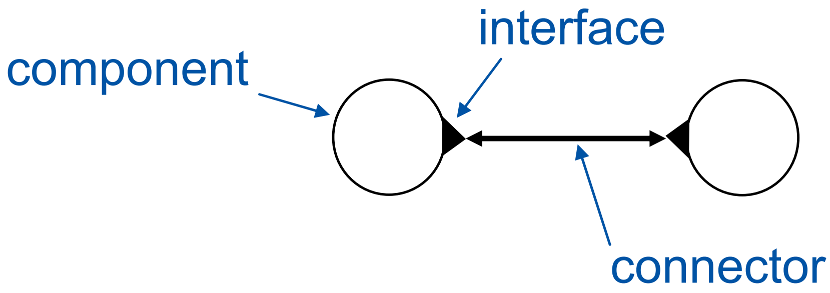
**I won't draw fixed distinctions between**

- Component
- Subsystem
- System

**Component-of is a relation, not an absolute.**

# Component & Architecture Based Development

Honeywell




**There are easy and dependable ways to**

- assemble components to create systems
- replace/add/remove components to modify systems

**We want**

- reusable components
- reusable architectures (patterns)

# Good Architectures



Provide stable interfaces between variable components

- localize complex dependencies within components
- localize technology change
- localize supplier dependencies

Are adaptable to multiple products

- Localize potential product upgrades
- localize variation between products
- mix-n-match components
- configurable components
- configurable & scalable architectures (patterns)

# Challenges



There is product-to-product variation in

- functional requirements
- vehicle characteristics
- dependability requirements
- cost and volume (NRE vs RC)
- physical environment (civil, defense, space, automotive)
- assurance/certification processes and regulations
- customer mandates
- legacies (is reuse always a good thing?)

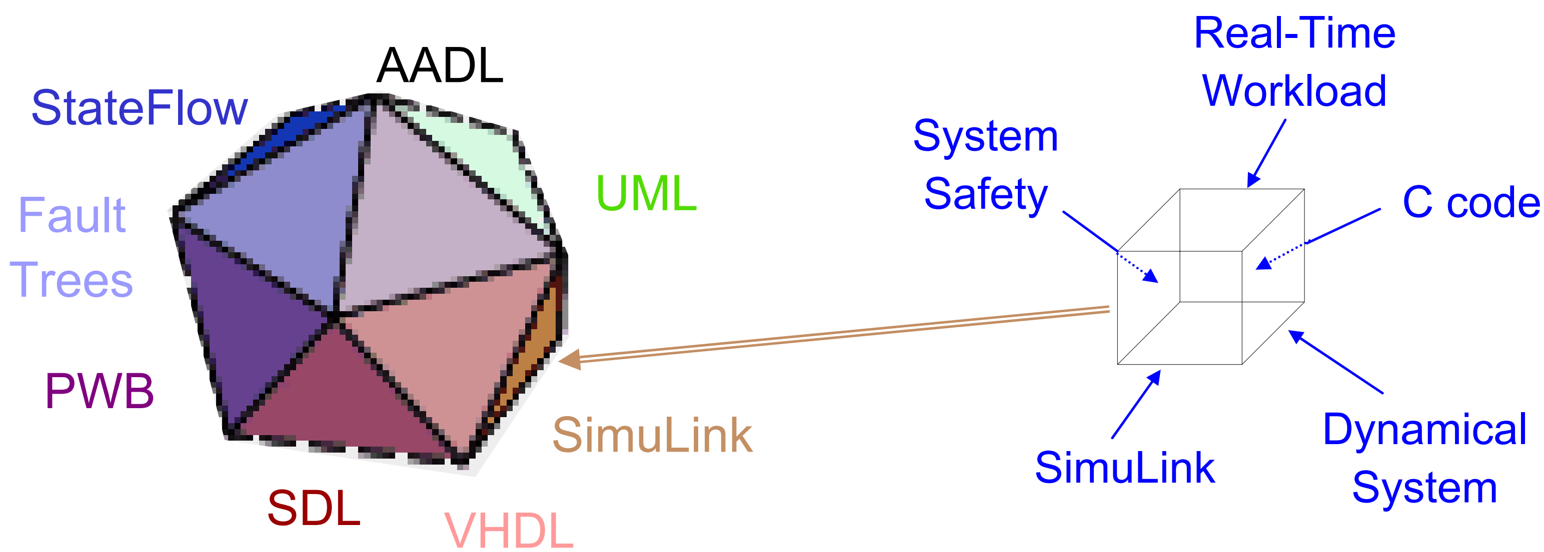
# Models



Systems are specified by multiple people trained in multiple disciplines using multiple notations and methods and tools.

A model is a specification written in a notation whose concepts, syntax, and semantics are drawn from a particular discipline and theory.

**There are always multiple related models for complex systems.**




**System** **Component**

# Comments on Models

Honeywell

**The set of models depends on the product line and organization**

**Models are neither completely dependent or independent, e.g.**

- WCETs in real-time model depend on hardware and software models
- C code must comply with AADL interface specifications

**Between multiple models we need**

- Mappings (traceability)
- Verifiable consistency relations
- Change propagation

**Practical desires**

- As many models as necessary, as few as possible
- As orthogonal as possible (separation of concerns)
- Meta-modeling tools with multi-model support
- Tool integration frameworks with multi-model support
- Automated traceability between models
- Automated change propagation between models

**Practical risks**

- Tower of Babel
- Plethora of hacked modeling languages

# Abstractions

Honeywell

During system development detail is progressively added, producing a series of increasingly detailed specifications.

- Sequence of specifications
- Refinement of a specification

Abstraction is a relation between a pair of specifications.

# Comments on Abstraction

Honeywell

Do we have good formalisms for abstraction?

- Hierarchical models
- Abstract interpretation provides safe tractable model checking.
- Weak equivalences (e.g. implements, conforms) provide assurance of live substitutability.
- Category theory can provide a semantics for refinement.
- Inherently underdetermined models, e.g.
  - Sets of logical properties
  - Hybrid systems without equality
  - Partial orders

Abstraction is most easily formally defined within the same theory or model, so abstractions are not distinct types of models.

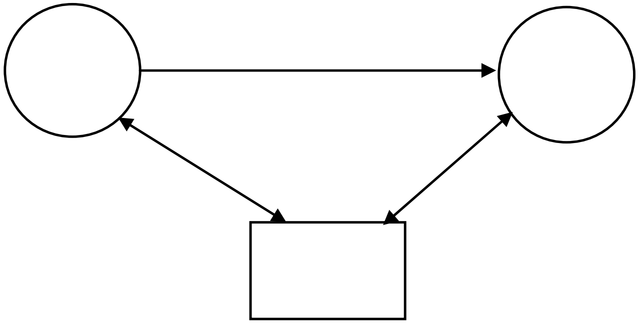
Practical desires

- Notations that support abstraction and refinement
- Intuitive relationship between human-created abstract and concrete
- Live and safe abstractions
- Verifiable compliance relations between abstract and concrete
- Frameworks that support traceability

# Architectures

Honeywell

An architecture is a static set of component specifications that are connected together in a specified way.



Architectures can be changed by

- Changing/refining a component or connection
- Adding or removing components or connections

A component specification appearing in an architecture diagram is often an abstraction, to be instantiated with a concrete component.

# Comments on Architectures

Honeywell

Ideally, assurance of component usability can be done locally:

correctness of architecture of abstract components

& compliance of concrete to abstract components

→ correctness of architecture of concrete components

Assumptions about the context of use (plant, environment) are sometimes used as part of the abstract component specification.

Compliance of concrete to abstract alone (e.g. implementation to interface) may not be sufficient.

Robustness is the degree to which a concrete component will work satisfactorily in the face of variability or uncertainty in the context of use.

Can we deal with robustness in a more methodical and formal way (e.g. as feed-back control people do)?

Robustness and abstraction may be related, e.g. the more abstract the context of possible use, the more robust the component.

# What is a component?

Honeywell

In fullest generality, a component may be

- A set of parametric (configurable) models
- Mappings between models
- Multiple levels of model abstraction
- A set of abstract context-of-use architectures

Robustness is a desirable component quality.

Configurable, abstract architectures (patterns) are desirable.

# Possibly Fruitful Research Topics

Honeywell

Relationships between multiple models

Abstraction

Abstract context-of-use architectures as part of abstract component specifications

Robustness

# Further Information



SAE AADL Information Site

http://www.aadl.info/

Architecture Analysis & Design Language (AADL) workshop and SAE standardization committee meeting, Paris, France, 17-21 October 2005.

Contact: Jean-Francois.Tilman@axlog.fr [+33]1 41 24 31 33

Kirk Schloegel, David Oglesby and Eric Engstrom, *Towards Next Generation Metamodeling Tools*, OOPSLA 2002, Seattle,

http://citeseer.ist.psu.edu/637526.html

Pam Binns and Steve Vestal, *Hierarchical Composition and Abstraction in Architecture Models*, WADL, Toulouse, 2004,

http://www.laas.fr/FERIA/SVF/WADL04/presentations.php