Modeling in an Architectural Variability Description Language

RuG: Theo Dirk Meijler, Silvie Schoenmaker

Thales: Egbert de Ruijter
Agenda

- Motivation
- Requirements for Architectural Variability Modeling
- Modeling Elements/ Basic Examples
- Validation
- Conclusion/ Future
Motivation: Line of Reasoning

Product Lines: large scale Reuse

Support for Product Configuration: efficient product derivation

Architectural Configuration: important view

Extension of Architectural Description Language With Explicit Variability: AVDL
Product Lines: Large Scale Reuse

- Building Software from Scratch: too expensive(!)
  - High Complexity
  - Reliability

=> Reuse is necessary

- From Libraries, via Frameworks and Components to Product Lines/ Product Families
Product Lines: Large Scale Reuse

- Pre-defined Application Architecture
  - E.g. layered, repository
- Use of Pre-existing Components
- Components with “in-built” variability

=> Reuse for a specific Application Domain
Support for Product Configuration: Efficient Product Derivation

- Variability is so large, that building specific applications becomes (again!) too difficult
  - Very Specific Knowledge (dependency on experts)
  - Thousands of Parameters (trial and error!)
    - Thales: avg. Of 3000 hrs parameterization
Architectural Configuration: “Important View”

Software Development is about describing “modeling” a system

Kruchten: “4+1” View of Software Development:
- Process View
- Logical View
- Deployment View
- Physical View

+1: The Scenario’s

Main Principle: Distinguish these Views for Configuration
Architectural Configuration:

“Important View”

Process View:

Run-time Components Working together through Connectors
(e.g., event channels, message queues..)

Which Components Work together and How!

ADL: Architectural Description Language

Needed: Explicit Variability, Configurability

\[\Rightarrow AVDL\]
Motivation: Line of Reasoning

Product Lines: large scale Reuse

Support for Product Configuration: efficient product derivation

Architectural Configuration: important view

Extension of Architectural Description Language With Explicit Variability: AVDL
Requirements for AVDL

- Clear Modeling Notation (e.g. like UML)
- Can be Formalized
- Variability and Choice in one Model
- Pinpoint Location of Variability:
  - “Where” in Architecture (Variation Points)
- Take into account Binding Time
- Different kinds of Variation Points
  - Optional, Alternative, (Optional) (Set) of Alternatives
Agenda

- Motivation
- Requirements for Architectural Variability Modeling
- Modeling Elements/ Basic Examples
- Validation
- Conclusion/ Future
Basic Elements

Legend:
- Component X of type Com1
- Interfaces Input Xin Output Yout
- Role of name Bout
- Connector Bus of type TBus
- Link
Simple Choice

Legend:
(see prev.)

- Link between variable role descriptor and (elements of) choice set
- Variable role (VR) descriptor for role Bin, variation point VP1, cardinality 1..1, Binding time AD,
Related Choices:
Architectural Variation Point

VR Bout and Bin are related by both corresponding to VP1
Variation in Sub-architecture

Legend:
(see prev.)

Sub-component X linked via IoIn to Io of component S

Variable Interface
Descriptor of interface SIn, Corresponding to Variation point VP-S,
Agenda

- Motivation
- Requirements for Architectural Variability Modeling
- Modeling Elements/ Basic Examples
- Validation: Bosch & Thales Examples
- Conclusion/ Future
Bosch Example (Interfaces)
Thales Example (Dependency)
Conclusion

- Motivation
- Requirements for Architectural Variability Modeling
- Modeling Elements/ Basic Examples
- Validation
- Conclusion/ Future
Conclusion/ Further Work

- Modeling Language Used Successfully for Industrial Examples

Further Work

- Mapping to CKML
- Connection to Variation Point Modeling
- A Tool! -> Further Validation