Faris M. Taweel’s PhD thesis: 
An Approach To The Definition of Domain-specific software component models

Michal Malohlava
DISTRIBUTED SYSTEMS RESEARCH GROUP
http://dsrg.mff.cuni.cz
CHARLES UNIVERSITY PRAGUE
Faculty of Mathematics and Physics
Motivation

• Domain-specific component-based systems
  ▪ To encapsulate domain knowledge
  ▪ Tools

• Producing families of software in CBSE
  ▪ Domain-specific CS describes a product family
• Domain modeling introduction

• Approach idea

• K. K. Lau’s component model short overview

• Process of deriving domain-specific component models

• Example
Domain modeling

• Notion of domain
  ▪ Defined by domain model in a context (defined via Context model)

• Domain model includes
  ▪ Functional model
    • Function model
    • Control model
  ▪ Feature model
    • variability
  ▪ …

• FODA
Domain model

- **Domain model** is a starting point for domain-specific development approaches
  - E.g. product lines, software factories

- => **domain model** often has to capture variability

- **Vision**: to derive a **domain-specific component-based model** from the **domain model**
What does the term “domain-specific CM” mean?

- Czarnecki: “Abstraction of the domain represents specific concepts or problems in that domain”
- I.e. knowledge of the domain is directly integrated into the model
  - E.g. SQL language in the area of DSL

Contemporary component models

- Domain-specific?
- No!

Why not?
- Koala (consumer electronics)
- Pecos (field devices)
- SaveCCM (vehicular systems)
Idea of domain-specific CM derivation

A generic Component Model

Domain Model  Mapping Process  Domain-specific Component Model

Domain Engineer Effort & Tools

• Notes
  ▪ Resulting domain-specific CM has to contain variation points to cover variability defined in domain model
  ▪ In fact, the idea follows the concept of designing DSLs
Idea of domain-specific CM derivation

A generic Component Model

Mapping Process

Domain-specific Component Model

Domain Engineer Effort & Tools

Domain Model
Idea of domain-specific CM derivation

A generic Component Model

Domain Model <-> Mapping Process <-> Domain-specific Component Model

Domain Engineer Effort & Tools
Generic component model (Lau)

- **Core concepts** – composition of *computational units* with help of *exogenous connectors*
  - **Computational unit (CU)**
    - Cannot call methods in other CU
  - **Exogenous connectors**
    - Encapsulates control => model control-flow

(a) Example.

(b) Control flow.
Generic component model (Lau)

- Connector types – help to define semantics
  - Atomic – CU + invocation connector (+guard)
  - Composite – Atomic components + connectors

Michal Malohlava, DSRG seminar, 4th November 2009
Idea of domain-specific CM derivation
Deriving Domain-Specific CMs

• **Input:** domain model
  - Context model
  - Feature model
  - Functional model
  - Function (data-flow) & control (control-flow) models
    - CT defined by FSM
Deriving Process

• Domain model analysis identifies & maps:
  ▪ Data transformations ($dt$) -> atomic & composite components
  ▪ Control transformations ($ct$) -> compositions operators (connectors)
  ▪ Data storage
  ▪ Extra-functional properties

• Analysis takes into account feature model of the domain to identify optional $dt$, $ct$
Deriving Process

- Identification of *atomic components*:
  - Primitive data transformation (*pdt*) performing I/O or computation
  - A group of *pdts* associated with a control transformation (+ constraints on only one pdt activation)
Deriving Process

• Identification of connectors
  ▪ Identification of CERs (control-encapsulation region)
    • CER = group of cts,dts which together encapsulate control
    • With or without feedback discussion
    • CERs are hierarchical (in the means of dts)
  ▪ CER is derived into a composite connector

Michal Malohlava, DSRG sen
Deriving Process

• Result

- Domain-specific model including
  - connector types (with variation points)
  - atomic components
- Both encapsulates domain knowledge
- Used for designing products defined by input feature diagram
Example of VSD - inputs

• Vehicular systems domain (VSD)
  ▪ Of course little bit simplified ;-

• Input: Domain model containing:
  ▪ Feature model – summarizes the requirements for possible product variants

```
VSD
  └── Auto Cruise Control
      ├── Cruise
      │   └── Object Detection
      │       └── Measure Motion
      │           └── Calibrate
      └── Monitoring
            └── Status

"Object Detection" requires "Cruise"
```

Michal Malohlavi
Example of VSD - inputs

- **Input:** Domain model containing:
  - Functional model
  - Context model defining external entities (e.g., sensors, actuators, clocks)
Example of VSD – derivation process

- Identification of extra-functional properties
- Identification of CERs
  - Depends if *optional features* ([3] – Control throttle, [5] – Object detection) are included
Examples of VDS – atomic components

- Decomposition of data transformations to primitive data transformation

5.1 Scan Signal
5.2 Calc. Speed

3.3 Maintain Accel
3.4 Adapt Speed

3.1 Select Speed
3.2 Maintain Speed
Examples of VDS - connectors

- Deriving connectors for CERs
  - Atomic component = \textit{pdt}

<table>
<thead>
<tr>
<th>Connector</th>
<th>$S_i$</th>
<th>Control Flow</th>
<th>DS Initialised</th>
<th>$S_f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_1$</td>
<td>Accelerating OR Cruising</td>
<td>BRAKING</td>
<td>ANY</td>
<td>Idle</td>
</tr>
<tr>
<td>$G_2$</td>
<td>Accelerating OR Cruising</td>
<td>TOP_GEAR=Off</td>
<td>ANY</td>
<td>Idle</td>
</tr>
<tr>
<td>$S_1$</td>
<td>Idle</td>
<td>RESUME</td>
<td>Initialised</td>
<td>\textit{pdt}_{3.2}</td>
</tr>
<tr>
<td>$S_2$</td>
<td>Cruising OR Idle</td>
<td>START ACCEL</td>
<td>ANY</td>
<td>\textit{pdt}_{3.3}</td>
</tr>
<tr>
<td>$S_3$</td>
<td>Accelerating</td>
<td>STOP ACCEL</td>
<td>NOT BRAKING</td>
<td>\textit{pdt}<em>{3.1}; \textit{pdt}</em>{3.2}</td>
</tr>
<tr>
<td>$G_3$</td>
<td>Idle</td>
<td>Activate = CRUISE</td>
<td>ANY</td>
<td>\textit{pdt}<em>{3.1}; \textit{pdt}</em>{3.2}</td>
</tr>
</tbody>
</table>
Example of VDS – design of CCS

- Design Cruise Control System
  - Resulting repository of model artifacts for designing VDS

<table>
<thead>
<tr>
<th>Connectors</th>
<th>Atomic Components</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector</td>
<td>Description</td>
<td>Variant</td>
</tr>
<tr>
<td><strong>CC₃</strong></td>
<td>Speed controller</td>
<td>base</td>
</tr>
<tr>
<td><strong>CC₃₀₁</strong></td>
<td>Speed Controller</td>
<td>Variant of <strong>CC₃</strong></td>
</tr>
<tr>
<td><strong>CC₃₋₅</strong></td>
<td>ACC controller</td>
<td>Variant of <strong>CC₃</strong></td>
</tr>
<tr>
<td><strong>CC₄</strong></td>
<td>Monitor Controller</td>
<td>base</td>
</tr>
<tr>
<td><strong>S</strong></td>
<td>Selector</td>
<td>base</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td>Pipe</td>
<td>base</td>
</tr>
</tbody>
</table>

Maintain Speed  \( p_{dt3.2} \)
Maintain Accel  \( p_{dt3.3} \)
Adapt Speed  \( p_{dt5.2} \)
Scan Signal  \( p_{dt5.1} \)
Measure Mile  \( p_{dt1} \)

----------

Michal Malohlava, DSRG seminar, 4th November 2009
Example of VDS – design of CCS

- The resulting model of cruise control system
  - Model product family
  - Variation point in CC3 – variants CC\textsubscript{3v1}, CC\textsubscript{3-5}

<table>
<thead>
<tr>
<th>Connectors</th>
<th>Atomic Components</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector</td>
<td>Description</td>
<td>Variant</td>
</tr>
<tr>
<td>CC\textsubscript{3}</td>
<td>Speed controller</td>
<td>base</td>
</tr>
<tr>
<td>CC\textsubscript{3v1}</td>
<td>Speed Controller</td>
<td>Variant of CC\textsubscript{3}</td>
</tr>
<tr>
<td>CC\textsubscript{3-5}</td>
<td>ACC controller</td>
<td>Variant of CC\textsubscript{3}</td>
</tr>
<tr>
<td>CC\textsubscript{4}</td>
<td>Monitor Controller</td>
<td>base</td>
</tr>
<tr>
<td>S</td>
<td>Selector</td>
<td>base</td>
</tr>
<tr>
<td>P</td>
<td>Pipe</td>
<td>base</td>
</tr>
</tbody>
</table>

Michal Malohlava, DSRG seminar, 4th November 2009
Issues

• No concurrency
  ▪ Component, connectors are passive elements

• Mapping to implementation model is not presented, no implementation (?)

• Resulting model is not readable without knowing semantics of AC, connectors
Challenges for ‘*-SOFA-*’

• Clarify domain models, domain modeling
  ▪ Meta-component system

• Computational model, semantics
  ▪ Definitely useful for SOFA HI

• Data flows & Control flows
  ▪ Entities can be seen as the first attempt to capture ‘data flows’
  ▪ Multiple model views (mentioned by Premek, INRIA guys)
Thanks for your attention!