CBSE and MDE: Fitting the Pieces Together

Kung-Kiu Lau

School of Computer Science
The University of Manchester
United Kingdom

kung-kiu.lau@manchester.ac.uk

Keynote, ModComp 2016, 4 October 2016, Saint-Malo, France
Overview

Structure of Talk
- MDA/MDE/CBSE: Terminology, essential elements and links
- Our work in CBSE
- Our use of MDE
- Observations/questions on MDE

Acknowledgement: Joint work with Simone di Cola and Cuong Tran
## MDA vs CBSE

<table>
<thead>
<tr>
<th>MDA</th>
<th>CBSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>process-centric</td>
<td>product-centric</td>
</tr>
<tr>
<td>top-down</td>
<td>bottom-up</td>
</tr>
<tr>
<td>correct-by-transformation</td>
<td>correct-by-composition</td>
</tr>
</tbody>
</table>

### Diagram:

- **Business Process**
  - CIM: Computation-independent Model
  - PIM: Platform-independent Model
  - PSM: Platform-specific Model

- **Transformations**:
  - Business Process to Business Functionality and Behaviour
  - Business Functionality and Behaviour to System Model
  - System Model to Code

- **System Model**:
  - (Libraries, Components, ...)

---

Kung-Kiu Lau (University of Manchester)
MDE

Meta-metamodel
e.g. Ecore, GME Metamodel

Metamodel
e.g. UML Metamodel

Model
e.g. UML diagrams

Transformation Language
e.g. ATL (M2M), Acceleo (M2T), GReAT (M2M)

Transformation Rules
e.g. UML to code (M2T)

Transformation execution

Implementation
e.g. executable model, code
MDE + CBSE: Our Approach

Meta-metamodel
- e.g. Ecore, GME Metamodel

Metamodel
- FX-MAN Metamodel

Model
- FX-MAN architecture

Transformation Language
- e.g. ATL (M2M), Acceleo (M2T), GReAT (M2M)

Transformation Rules
- FX-MAN to functional model (M2M)
- FX-MAN to code (M2T)

Transformation execution

Model
- Functional model (Domain model)

Implementation
- Executable FX-MAN system

MDE + CBSE with Component Model (FX-MAN)
- Repository = Pre-existing components (in a domain)
- Repository components reused in many systems (in the domain)
- System = Composition of components
- Composition $\rightarrow$ Reuse
- ‘Bottom-up’
## Components and Composition

<table>
<thead>
<tr>
<th>Unit of Composition</th>
<th>Composition Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Containment</td>
</tr>
<tr>
<td>Function</td>
<td>Function nesting</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Procedure</td>
<td>Procedure nesting</td>
</tr>
<tr>
<td>Class</td>
<td>Class nesting</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Mixin</td>
<td>Mixin inheritance</td>
</tr>
<tr>
<td>Mixin/Class</td>
<td>Mixin-class inheritance</td>
</tr>
<tr>
<td>Trait</td>
<td>Trait composition</td>
</tr>
<tr>
<td>Trait/Class</td>
<td>Trait-class composition</td>
</tr>
<tr>
<td>Subject</td>
<td>Subject composition</td>
</tr>
<tr>
<td>Feature</td>
<td>Feature composition</td>
</tr>
<tr>
<td>Aspect/Class</td>
<td>Weaving</td>
</tr>
<tr>
<td>Module</td>
<td>Module nesting</td>
</tr>
<tr>
<td>Architectural unit</td>
<td></td>
</tr>
<tr>
<td>Fragment box</td>
<td>Invasive composition</td>
</tr>
<tr>
<td>Process</td>
<td></td>
</tr>
<tr>
<td>Web service</td>
<td></td>
</tr>
<tr>
<td>Encapsulated component</td>
<td></td>
</tr>
</tbody>
</table>

A software component model defines:

- components
- composition mechanisms

CBSE with a component model is model-driven by definition:

- model for components
- model for composition
  + model-driven implementation of components
  + model-driven implementation of composition


Types of Components

A Generic Component

Required Service ——— Provided Service

An Object

Provided method

An Architectural Unit

in1 ——— out1
in2 ——— out2

An Encapsulated Component

Kung-Kiu Lau (University of Manchester)
Types of Composition Mechanisms

Connection: Method Call & Port Connection

(a) Direct message passing

(b) Indirect message passing

Coordination: Exogenous Composition

Communication channel
Idealised Component Life cycle
Composition in Component Design Phase and Component Deployment Phase

Idealised Component Life Cycle

Design Phase

- Builder
- Repository

Deployment Phase

- Assembler

Run-time Phase

- Environment

Component (source code) Component (binary) Component instance


Traditional CBSE Desiderata

<table>
<thead>
<tr>
<th>Desideratum</th>
<th>Design Phase</th>
<th>Deployment Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Components should pre-exist</td>
<td>Deposit components in repository</td>
<td>Retrieve components from repository</td>
</tr>
<tr>
<td>Components should be produced independently</td>
<td>Use builder</td>
<td>—</td>
</tr>
<tr>
<td>Components should be deployed independently</td>
<td>—</td>
<td>Use assembler</td>
</tr>
<tr>
<td>It should be possible to copy and instantiate components</td>
<td>Copies possible</td>
<td>Copies and instances possible</td>
</tr>
<tr>
<td>It should be possible to build composites</td>
<td>Composition possible</td>
<td>Composition possible</td>
</tr>
<tr>
<td>It should be possible to store composites</td>
<td>Use repository</td>
<td>—</td>
</tr>
</tbody>
</table>

**Taxonomy of Component Models**

Category 1: Design without Repository (Acme–like ADLs, UML2.0, PECOS)

Category 2: Design with Deposit–only Repository (EJB, OSGi, Fractal, COM, .NET, CCM)

Category 3: Deployment with Repository (JavaBeans, Web Services)

Category 4: Design with Repository (Koala, SOFA, KobrA, SCA, Palladio, ProCom)

Category 5: Design and Deploy with Repository (X-MAN)

Present

Taxonomy of component models shows:
- Current component models do not fully meet the traditional CBSE desiderata

Future

- CBSE faces new challenges:
  - increased scale
  - increased complexity
  - assurance of safety of large complex systems

- Future component models have to meet these new desiderata
Hierarchical (algebraic) composition \( \rightarrow \) scale and complexity

Compositional verification \( \rightarrow \) large-scale verification (of safety)


Vehicle Control System (VCS)

A VCS is a real-time, on-board system for supervising a vehicle. It manages several routine services and tasks, including:

- **statistical data calculation**
  
  e.g. of fuel consumption and of average speed

- **observation or monitoring of the vehicle’s internal state**
  
  e.g. maintenance status

- **cruise control**
  
  i.e. automatically controlling the vehicle’s speed in such a way that a steady (cruise) speed can be set (by the driver) and then maintained by taking over control of the throttle whenever necessary

- **collision detection**
  
  to ensure safety and enable automatic driving (while cruising)
VCS System in X-MAN

[Diagram of a system with nodes and connections, labeled with components such as AverageMPH, AverageMPG, Monitoring, AutoCruiseControl, and AllRoundDetection.]

Kung-Kiu Lau (University of Manchester)
VCS Functional Model: State Chart
Context for CBSE is a **domain** (of multiple systems)

- Separate life cycles for components and systems
Compositional V & V

The W Model

Component Life Cycle

Domain knowledge

Component design

Coding

Component V&V & certification

System requirements

System specification

Component selection & adaptation

System assembly

Coding

System V&V

Acceptance testing

**Domain Model** = Feature Model + Functional Model (Behaviour)

**Domain Engineering** = Domain Knowledge $\implies$ **Domain Model** $\implies$ **Product Family Architecture** (Reference Architecture)

FX-MAN Example: Family of ECL Products

Feature Model

386 product variants (28688 without constraints)
FX-MAN Example: Family of ECL Products

Product Family Architecture

386 product variants (28688 without constraints)
### Models and Transformations

<table>
<thead>
<tr>
<th>Component Model</th>
<th>Model</th>
<th>M2M</th>
<th>Model</th>
<th>M2T</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-MAN</td>
<td>Functional Model</td>
<td>←</td>
<td>Component</td>
<td>→</td>
<td>Code</td>
</tr>
<tr>
<td>X-MAN</td>
<td>Functional Model</td>
<td>←</td>
<td>Component Composition</td>
<td>→</td>
<td>Code</td>
</tr>
<tr>
<td>FX-MAN</td>
<td>Product Family</td>
<td>Variation operator</td>
<td>Product Family</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FX-MAN</td>
<td>Functional Model</td>
<td>←</td>
<td>Family Composition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FX-MAN</td>
<td>Functional Model</td>
<td>←</td>
<td>Product Family Architecture</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Functional Model = State Charts + Activity Charts
Product Family = Set of X-MAN Architectures

### Technology Stack

- Spray
- Graphiti
- Xtend + Acceleo
- Xcore (EMF)
- CDO
What We Have Done

- We use models everywhere
- We use MDE for tool development
- We have not focused on platforms, or associated M2M transformations

Tool Downloads

http://www.click2go.umip.com/i/software/x_man.html

Feedback most welcome!
Some Observations/Questions on MDE Technology

- More modelling elements?
  - composition (not just association and containment)
  - components (units that are more compositional than classes)
  - behaviour (e.g. control, coordination)

- Higher-level abstractions?
  - not just classes
  - less coupled to OO technology
  - more hierarchical modelling (more than referencing)
  - model transformations may be challenging