Aspect Oriented Modeling: from Requirements to Models@Runtime

Prof. Jean-Marc Jézéquel
jezequel@irisa.fr
http://www.irisa.fr/prive/jezequel

Outline

• AOMDE\(^2\)
  - mete-meta issues of Aspect Oriented Model Driven Engineering
  - Tools to build tools to build software: Kermeta

• AOMDE\(^1\)
  - meta issues of Aspect Oriented Model Driven Engineering
  - A tool to build software: SmartAdapter, a generic model weaver

• AOMDE\(^0\)
  - issues of Aspect Oriented Model Driven Engineering
  - Building software: using SmartAdapter in DiVA for weaving models at runtime

• Demo of the DiVA Studio
Naive Model Driven Engineering

• Functionality
• Security, survivability, robustness
• Safety
• Fault tolerance

Modelling activity

Code Model

Aspect Oriented Software Development

• Large and complex systems compose many different concerns
  ▪ This affects modularization
  ▪ Need to manage interactions between concerns
• Scattering
  ▪ One concern distributed all over the system, not put in a well identified unit
• Tangling
  ▪ One unit contains elements from different concerns
Aspect Oriented Modeling

- Separation of concerns in a model
  - A concern is not necessarily cross-cutting
- Model composition
  - Build a global view from a set of concern models
- AOM is a wide domain
  - Captures a large variety of modeling practices
  - A large number of composition approaches

Aspect Oriented Model Driven Engineering

AOMDE = Pleonasm because a Model is an Abstraction of an Aspect of Reality for a given Purpose

Change one Aspect and Automatically Re-Weave: From AORE, SPL to DAS
Naive MDE vs (AO)MDE

- Transformation vs. Composition
  - Similar to *goto* vs proper *loop* concept in language
  - Or assembly language vs. High-Level Languages

\[(\text{AO}) \text{ MDE} = (\text{AO}) \text{ Modeling} + \text{Composition}\]

AOM: Why?

• Intent of separation of concerns
  ▪ Handle complexity by decomposition
  ▪ (Dynamic) Product line modeling: features and variation points are modeled as separate concerns
  ▪ Reusable aspect models: build models that can be reused in the design of different systems
  ▪ Analyzable concerns: separate the characteristics of a system in order to analyze them separately before building a larger system

Composition: Why?

• Collaborative development: compose models that have been developed in parallel
• Compose different variants to limit the maintenance cost
• Analyze the result of composition
• Use the result of composition as a new model
AOMDE: When?

• Separate concerns at different stages
  ▪ Requirements engineering: identify features and cross-cutting concerns from requirement documents
  ▪ Feature modeling: AOM for product derivation
  ▪ Architecture
  ▪ Design
  ▪ Runtime (for system dynamic adaptation)
• Implies different techniques for composition

Composition: what?

• Identify the "similar" elements in both models
• Elements are "similar" in two models if they have the same "meaning"
Composition: what?

- Difficult to establish the interpretation relation for each element in the model
- A little bit easier to compose elements from the same metamodel
  - Only elements that have the same type can have the same interpretation
  - When the models to compose have different metamodels it is necessary to specify interpretation at the meta level

Composition: where?

- Critical for behavioural models
  - When composing scenario A after B does not mean the same as B after A
- The place where the elements should be composed (joinpoints) can be declared with a pattern language
  - To define predicates (pointcuts) over a model
    - Mata, Smartadapters, RAM
Composition: how?

- What process to perform on the model to integrate new elements
  - Merge, insert, replace, etc.
- Default strategies in some composition algorithms
  - Match and merge, signature-based
- Experience shows that explicit strategies are often needed
  - Cf. Kermeta/Smartadapters

Model Composition - Scheme

Kernel Meta-modeling Language: Kermeta

A tool to build tools to build software

- Kermeta is a Model-Oriented Language
  - based on an AO/OO executable meta-modeling paradigm
    - Static typing, generics, functions objects, reflection…
  - First language where models are first class entities
    - Allows interesting questions to be asked; e.g.; what is the type of a model?
- Executable meta-modeling allows:
  - specification of abstract syntax, static semantic (OCL) and dynamic semantics, connection to the concrete syntax.
  - model and meta-model simulation and prototyping
  - model transformation, design level aspect weaving
Kermeta: Breathing life into Meta-Models

// MyKermetaProgram.kmt
// An E-MOF metamodel is an OO program that does nothing
require "StateMachine.ecore" // to import it in Kermeta

// Kermeta lets you weave in aspects
// Contracts (OCL WFR)
require "StaticSemantics.ocl"
// Method bodies (Dynamic semantics)
require "DynamicSemantics.kmt"

// Transformations

aspect class FSM {
operation reset() : Void {
    currentState := initialState
}
}

class Minimizer {
    operation minimize (source: FSM): FSM {…}
}

OO Frameworks developed with Kermeta

• Many Model Development Kits
  ▪ Aka Kermeta code « tooling » specific MM
    • UML2, EMOF, Java5, Kermeta, Kompose…

• General purpose Tools
  ▪ Model Typing/Generic Refactorings/Metrics
  ▪ AOM/Composition tools
    • Kompose (R. France, CSU)
    • GeKo (J. Klein, U. Luxemburg & J. Kienzle, McGill)
    • SmartAdapters (P. Lahire, U. Nice)…

• A Kermeta compiler is available to deliver JARs without any dependency with Kermeta
Outline

• AOMDE²
  ▪ meta-meta issues of Aspect Oriented Model Driven Engineering
  ▪ Tools to build tools to build software: Kermeta

• AOMDE¹
  ▪ meta issues of Aspect Oriented Model Driven Engineering
  ▪ A tool to build software: SmartAdapter, a generic model weaver

• AOMDE⁰
  ▪ Issues of Aspect Oriented Model Driven Engineering
  ▪ Building software: using SmartAdapter in DiVA for weaving models at runtime

• Demo of the DiVA Studio

Smartadapters: a tool to build software

• A generic framework for AOM
  ▪ Built with Kermeta (a tool to build tools)
  ▪ Can be adapted to different modeling languages

• Intends to be as flexible as possible

• P. Lahire, B. Morin, G. Vanwormhoudt, A. Gaignard, O. Barais, and J.-M. Jézéquel. « Introducing variability into aspect-oriented modeling approaches ». MoDELS’07
Weaving Process

- **Two-phased:**

1. Detection of the join points

   Use a generic Prolog-based pattern-matching implemented in Kermeta.

   \[\text{[Ramos, Barais, Jézéquel@MODELS2007]}\]

   -> yield a list of join points

2. Generic Composition of the Advice at the level of the join points.

---

Composition Details

The morphisms give Built-in Tracability for free!
Smartadapters: 3 elements for composition

Join points identification

Advice
Add a root element
Add a final state

Composition

Smartadapters: a generic composition framework

Formally, the Pointcut Language MM is the topmost supertype of the base language MM containing all of its concrete concepts.

Domain-specific AOM framework

1. Pointcut language
2. Domain-specific adaptations

Generic

Pattern matching engine
Adaptation metamodel + weaver

uses
automatic generation
customization
Generating the Pointcut Language

$\text{MM}' = \text{MM except:}$

- No invariant or precondition in $\text{MM}'$
- All features are optional in $\text{MM}'$ (lower bound:=0)
- No abstract element in $\text{MM}'$

Formally, $\text{MM}'$ is the topmost supertype of $\text{MM}$ containing all of its concrete concepts
Generating Adaptations

Composing this snippet means:
- adding final and t in the base FSM
- setting the source of t

2. For each metaclass MyMetaClass in MM
   - setMyMetaClass
   - unsetMyMetaClass
   - createMyMetaClass
   - cloneMyMetaClass

- setFSM unsetFSM createFSM cloneFSM
- setTransition unsetTransition createTransition cloneTransition
- setState unsetState createState cloneState
Smartadapters: example

Template (pointcut)

SetFSM
- aFSM: anyFSM
- states: {final}
- transitions: {t}

setTransition
- aTransition: t
- aSource: any
- aTarget: final

makeUnique
- element: final

No aspect/base coupling
protocol = f(template, structure)

Weaving engine

- Load adapter + base model
- Pattern matching: {bindings}
  - Binding: Role (template elt) -> base model element

- For each binding b selected by the user
  - Apply the composition protocol
    - Just call adapter.apply(b)
      (directly implemented in the adaptation metamodel)

- Save the result
Outline

- **AOMDE\(^2\)**
  - Meta-meta issues of Aspect Oriented Model Driven Engineering
  - Tools to build tools to build software: Kermeta

- **AOMDE\(^1\)**
  - Meta issues of Aspect Oriented Model Driven Engineering
  - A tool to build software: SmartAdapter, a generic model weaver

- **AOMDE\(^0\)**
  - Issues of Aspect Oriented Model Driven Engineering
  - Building software: using SmartAdapter in DiVA for weaving models at runtime

- Demo of the DiVA Studio

---

Application of SmartAdapter to DiVA: Dynamic Variability in complex, Adaptive systems

- 7th Framework Programme, funded by the EU (5 M€)
- **Partners:**
  - SINTEF Norway (research partner)
  - Lancaster University (research partner)
  - INRIA (research and technology partner)
  - pure-systems GmbH (technology partner)
  - Thales Therésis (application and service provider)
  - CAS Software ((application and service provider)
- **Duration:** February 1st 2008-January 31st 2011
Context aware software systems able to adapt automatically to changes in their environments

Towards more complex DAS

- Dynamic Adaptive Software (DAS) development:
  - Adaptation logic often embedded into application logic
  - Adaptation logic hard-coded using low-level APIs
  - Readability, maintainability, and communication with other stakeholder not easy

- Exponential growth of possible configurations
  - Convergence with Dynamic Software Product Lines
  - N features, N tending to be larger and larger
    - $2^N$ potential program configurations, $2^N \times (2^N-1)$ transitions
Technical Approach

- Separating the application-specific functionality from the adaptation concerns in the requirements
- Aspect-oriented techniques used to analyse and reconfigure crosscutting features dynamically
- Model driven techniques used to raise the level of abstraction by providing models at runtime, model composition and automatic reconfiguration of platform
Example

Refining variants (features) with aspect models

- Mandatory elements → Base model
- One variant (leaf feature) → One aspect model
- An aspect model
  - Is a fragment of architecture (What? = advice)
  - Should be easily plugged into the base architecture
    - Where? = pointcut
    - How? = weaving directives
CAS architecture: driving aspect

Base Model (Mandatory elements)
Base + Driving Aspect

Base + Driving + Smartphone
Aspect Weaving and Validation

- N aspects $\rightarrow 2^N$ possible programs
  - Each aspect can be woven or not
  - However, there are some constraints
- Design-time validation
  - As much as possible, but not always possible due to combinatorial explosion
  - Evolution of requirements, once the system is deployed prevent 100% beforehand validation
- Complemented with runtime validation
  - Invariant checking, simulation, etc
  - Performed on the model, not on the running system
  - Possibly performed outside of the running system

Generating reconfiguration scripts

![Diagram of Architecture Metamodel and reconfiguration process]
Office -> Driving + SmartPhone

Reconfiguration Script: automatically generated

Benefits of DiVA

- (D)SPL approach to tame
  - The combinatorial explosion of configurations
  - The quadratic explosion of transitions
- AOM to automatically build configuration
  - Runtime validation before adapting the running system
  - Simple roll-back
- MDE to automate reconfiguration
  - Generation of safe reconfiguration scripts
Models@runtime

- Aspect as variability units raised at the model level
  - No explicit representation of ALL possible configurations
  - Configurations obtained by weaving most adapted aspects on demand at runtime

- MDE for automation of model composition
  - Model Based Validation, Generation of reconfiguration scripts

- Applications
  - DiVA: Airport Crisis Management, CRM
  - Home Automation for Dependent Persons
    - Work with Rennes Metropolis, AFAP, Deltadore… Demo available

Conclusion

- AOM = (Model Composition)\(^{-1}\)
- AOMDE is what MDE is really about
- Thanks to Kermeta this is not just meta-bla-bla
  - A tool for building tools for building software
    - Eg an aspect weaver
      - For weaving aspect at runtime to handle safe dynamic adaption of complex system
  - It works for real!
• http://aosd.net/2010

New modularity concepts, methods and techniques for software systems

Feel free to participate to the many associated workshops!