Hyper-Agility: Handling Variability from Design-Time to Runtime

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

Outline

• The Need for Hyper-Agility

• From Software Product-Lines (SPL) … … to Dynamic Adaptive Systems (DAS)

• (Aspect-Oriented) Modeling of DAS

• Dynamic Adaption with Aspects & Models
Context aware software systems

- able to adapt to changes in their environments automatically

Agile Manifesto

- **Manifesto for Agile Software Development**
  - **Individuals and interactions**
    - over processes and tools
  - **Working software**
    - over comprehensive documentation
  - **Customer collaboration**
    - over contract negotiation
  - **Responding to change**
    - over following a plan
Towards Hyper-Agility

• Think of it as the Agile Manifesto @ runtime
  ▪ Individuals and interactions
  ▪ Working software
  ▪ Customer collaboration
  ▪ Responding to change

Example Application Domain

• Home-automation to help disabled people stay home
  ▪ Aging society
  ▪ Hospital have limited resources, rooms, etc
    • → Very short stays
  ▪ Long stays very expensive for people and society
  ▪ Houses, flats, etc should be equipped

• How do we produce a program for each of them:
  ▪ Individuals and interactions
  ▪ Working software
  ▪ Customer collaboration
  ▪ Responding to change
Entimid

Cf demonstration at « fête de la science »

EnTiMid: Integrating IoS & IoT
[ServiceWave '09]

Controlling Boiler/Shutters from GoogleAgenda through a browser or an iPhone

- OSGi based
- MDE centric
- Generative Approach
Many Different Needs 1/2

Mrs. Dupont

- Living at home
- Motion troubles
- Memory loss
- Speaks French (only)
- Home equipped with:
  - LonWorks (lights)
  - Velux (shutters)

Many Different Needs 2/2

Mr. John Doe

- English student
- Living at home
- He had an accident
- He likes technology
- Wheelchair equipped with remote access for:
  - Lights and shutters (KNX)
  - Multimedia (UPnP)
Their needs

Both
Medical/technical staff should be able to
• Check their health state
• Check home configuration (shutters, lights, heaters...)

Mrs. Dupont
Some daily tasks should be automated (motion troubles) or reminded (memory loss).

Mr. Doe
Would like to control everything remotely, with a unified protocol

Software Product Lines (SPL)

• Families of related software products
• The products have a common part and a variable part

Variable features:
• color
• input method
• radio
• Internet calling

Common features
• MMS
• GPS
• TCP/IP
• FM Radio

Nokia Product Line

Nokia 3710
• color = {plum}
• keypad, large keys
• MMS
• GPS
• TCP/IP
• FM Radio

Nokia 5230
• color = {white, black}
• touch screen
• MMS
• GPS
• TCP/IP
• FM Radio

Nokia 5800
• color = {red, blue, black}
• keypad, touch screen, stylus
• MMS
• GPS
• TCP/IP
• FM Radio

Nokia N800
• color = {black}
• full keyboard, touch screen, stylus
• MMS
• GPS
• TCP/IP
• Internet calling
Outline

• The Need for Hyper-Agility

• From Software Product-Lines (SPL) … … to Dynamic Adaptive Systems (DAS)

• (Aspect-Oriented) Modeling of DAS

• Dynamic Adaption with Aspects & Models

Different variability dimensions

➢ Protocols
  ▪ Low-level protocols: KNX, X2D, X10, etc
  ▪ High-level protocols: http, UPnP, DPWS, etc

➢ Devices
  ▪ Lights, heaters, shutters, etc

➢ Web services
  ▪ GoogleAgenda, skype, WeatherForecast, MSN
  ▪ PaaS: Nurse as a Service, Food Delivery, Ooshop

➢ Adaptation to handicap/current health state
  ▪ Motion, memory, perception, etc

➢ *Dynamic reconfiguration => DAS*
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

Challenges

- Explosion of the number of possible configurations of Entimid
  - \(10^{14}\) possible configurations! \(\rightarrow\) \(10^{28}\) transitions!

- Dynamic Adaptation
  - Evolution of the handicap
  - Houses should be configured remotely
    - No wires to connect/disconnect in the walls
  - No service interruption
    - Rebooting the system cannot be a solution (lives depend on the system)

- Reliability
  - Safe migration path
    - from a valid configuration to another valid configuration
  - Performance issue (time) not critical
Related works

**Reliability, Validation**
- K. Czarnecki et al. GPCE’06
- F. Fleurey et al. Models@Run.Time’08
- J. Whittle et al. MODELS’07
- B. Cheng et al. ICSE’06, AOSD’09
- E. Figueiredo et al. ICSE’08

**Variability Management**
- S. Appel et al. ICSE’06
- M. Mezini et al. FSE’04
- Hallsteinsen et al. Computer’08
- B. Morin et al. MODELS’08
- P. David et al. SC’06

**Dynamic Adaptation**
- Oreizy et al. ICSE’98
- OSGi, Fractal, OpenCOM, etc

---

**Validation VS Variability management**

- How to validate DAS?
  - Specify everything!
    - all the configurations: >10^{14}
    - all the transitions: ~10^{28}
  - Model checking, code generation

- Problems
  - Explosion: Time consuming, error-prone
  - Evolution of the system (not predicted)
    - Stop all -> Evolve the specifications -> model check
    - -> re-generate -> re-deploy
How to manage dynamic variability?

- Do not focus on configurations!
  - Write reconfiguration scripts, encapsulating « features »
- Depending on the context and/or user needs
  - Choose the most adapted scripts
  - Executes all the selected scripts to dynamically adapt the system

Problems

- Scripts written by hand (calls to reconfiguration API)
- Interactions, dependencies between scripts?
- Does the configuration (after executing scripts) make sense?
  - Hopefully yes…

Hyper-Agility with a DSPL Approach

- Focus on variability, not on configurations
- Build (derive) configurations when needed
  - JIT, On demand at runtime, caching…
- Validate configurations before actual adaptation
- Automate the reconfiguration process
Outline

• The Need for Hyper-Agility

• From Software Product-Lines (SPL) … … to Dynamic Adaptive Systems (DAS)

• (Aspect-Oriented) Modeling of DAS

• Dynamic Adaption with Aspects & Models

Why modeling: master complexity

• Modeling, in the broadest sense, is the cost-effective use of something in place of something else for some cognitive purpose. It allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose.

• A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. This allows us to deal with the world in a simplified manner, avoiding the complexity, danger and irreversibility of reality.

Jeff Rothenberg.
Naive Model Driven Engineering

- Reliability
- Security, survivability, robustness
- Safety
- Fault tolerance
- Functionality

Modelling activity

Code Model

Aspect Oriented Model Driven Engineering

- Distribution
- Security
- Functional behavior
- Fault tolerance
- Use case model

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

Design Model

Code Model

Book state : String

User: borrow, return, deliver, setDamaged, reserve
(AO) MDE

= 

(AO) Modeling 
+
 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 analyse 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

Weaving Aspects: SmartAdapters

- **Two-phased:**
  1. Detection of the join points
     - Uses Kermeta+DROOLS for pattern-matching
     - -> yield a list of join point
  2. Generic Composition of the Advice at the level of the join points.

- **SmartAdapters:** a generic framework for AOM
  - Built with Kermeta (a tool to build tools)
  - Can be adapted to different modeling languages

Weaving in AOM

The morphisms give built-in Tracability for free!
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.

1. **Pointcut language**

2. **Domain-specific adaptations**

**Generic**

- Pattern matching engine
- Adaptation metamodel + weaver

**Domain-specific AOM framework**

uses
automatic generation
customization

Smartadapters: 3 elements for composition

Join points identification

Advice
Add a root element
Add a final state

Composition

ClassA → ClassB

ClassA

ClassC

ClassB

ClassC
Generating the Pointcut Language

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

Formally, MM4 is the topmost supertype of MM containing all of its concrete concepts

Matching Model Snippets
R. Ramos, O. Barais and J.M. Jézéquel
accepted at the MoDELS’07 conference,
Sept. 30 to Oct. 5, Nashville, TN, USA
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

AnyFSM

False positive

Any

Template
(pointcut)

SetFSM
- aFSM: anyFSM
- states: {final}
- transitions: {t}
setTransition
- aTransition: t
- aSource: any
- aTarget: final
makeUnique
- element: final
No aspect/base coupling
protocol = \( f(\text{template}, \text{structure}) \)

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 \( \text{adapter.apply}(b) \)
      (directly implemented in the adaptation metamodel)
- Save the result
Outline

• The Need for Hyper-Agility

• From Software Product-Lines (SPL) … … to Dynamic Adaptive Systems (DAS)

• (Aspect-Oriented) Modeling of DAS

• Dynamic Adaption with Aspects & Models

Big picture

Design-time Validation

Device controllers

Runtime Validation

Target configuration

Derivation AOM

Light

Shutter

Metamodel

Model @runtime

Running system

Script Generator
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

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 constraint

- 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

Extensive design-time validation

- Still possible to validate everything, for small systems
  - Produce all the possible configurations by aspect weaving
  - Validate all the configurations

- Discussion
  - Time/resource consuming
  - The number of configurations explodes
  - … but they are automatically generated, by aspect composition

- Not scalable
Validation of aspect models

- **Aspect-Oriented Modeling**
  - Validate the DSPL at design-time
  - Strong theoretical background (graph theory)
  - Modular reasoning
  - \( \rightarrow \) interactions and dependencies detection
    - Using Critical Pair Analysis
    - \( \rightarrow \) weaving order

Two interacting aspects

- Device Proxy
- Light Filter
- Light1
- Shutter1
- require I18N
Two dependent aspects

- Critical Pair Analysis has limitations
  - Aspect1, Aspect2 $\rightarrow$ OK
  - Aspect1, Aspect3 $\rightarrow$ OK
  - Aspect1, (Aspect2, Aspect3) $\rightarrow$ ?

- Need to validate woven configurations
  - At runtime, when they are produced
Checking configurations at runtime

- **Focus on one configuration**
  - Not the whole dynamically adaptive system

- **Efficient roll-back**
  - The running system is not yet adapted
  - Just discard invalid models
  - Report to user

Generating reconfiguration scripts

[Diagram showing the process of generating reconfiguration scripts]

- Architecture Metamodel
  - component, port, binding, etc
  - invariants

- Source model
  - Introspection listeners

- Target model
  - Conforms to

- The running system
  - Analyzer

- Analyzer
  -Delayed synchro
  -transfo
  -adapts
Office -> Driving + SmartPhone

Reconfiguration Script: automatically generated

Wrap-up

• The Need for Hyper-Agility

• From Software Product-Lines (SPL) … … to Dynamic Adaptive Systems (DAS)

• (Aspect-Oriented) Modeling of DAS

• Dynamic Adaption with Aspects & Models
Handling Variability in DAS

- (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
    - Spin-off from INRIA/U. Rennes to leverage this technology…
      - Morin et al., Models@runtime, IEEE Computer 10/2009
Conclusion

• Aspects to model variability in DAS
• 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!