




# An MDA-Based Approach for Behaviour Modelling of Context-Aware Mobile Applications



Laura Maria Daniele


TTT 11 February 2009

## Outline

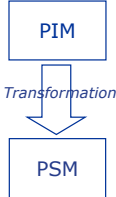
- ▶ MDA
- ▶ Behaviour modelling in MDA
- ▶ Our approach for behaviour modelling
- ▶ Case study
- ▶ Conclusions
- ▶ Future Work

11 February 2009 TTT 2




## Model-Driven Architecture (MDA)

- ▶ MDA aims at facilitating (distributed) systems design through the separation of **platform-independent** and **platform-specific** concerns
- ▶ Common pattern for MDA:
  - ▶ Define a platform-independent model (PIM)
  - ▶ Apply transformations to this model (model transformation)
  - ▶ Obtain one or more platform-specific models (PSMs)



11 February 2009 TTT 3




## Behaviour modelling in MDA(1)

- ▶ The MDA community agrees on the need to consider behavioural aspects in model transformations
  - ▶ No agreement on how this should be done

**Problem**

- ▶ Much attention to structural aspects in PSMs and generating code
- ▶ Less attention to the PIM level and behaviour of the modelled applications

11 February 2009 TTT 4




## Behaviour modelling in MDA(2)

**Consequence**

- ▶ Application behaviour is not (well) defined at the PIM level
- ▶ Behavioural aspects have to be incorporated later in the development process
  - ▶ Hand-written code to PSMs or to implementation code skeletons

11 February 2009 TTT 5



## Behaviour modelling in MDA(3)

**Our Solution**

An MDA-based approach for behaviour modelling of context-aware mobile applications

- ▶ This approach integrates behavioural aspects of the modelled applications at the PIM level
  - ▶ The PIM level is decomposed in different models
  - ▶ Each model is a refinement of the previous one

11 February 2009 TTT 6

University of Twente  
The Netherlands

### Approach

The diagram illustrates the service design approach. At the top, a tree structure shows actions (action1, action2, action3) leading to a **service specification (A-MUSE DSL)**. This specification is transformed into a **service design refined model (A-MUSE DSL)**. This refined model is then mapped to a **service design component model (ISDL)**, which consists of four components: ComponentA, ComponentB, ComponentC, and ComponentD. The process concludes with **platform selection**, leading to a **PSM** (Platform Specific Model).

11 February 2009      TTT      7

University of Twente  
The Netherlands

### Example: Service Specification

This diagram shows a service specification with several classes and their relationships. Key elements include:
 

- ServiceSpecification**: Contains methods like `me.getName()`, `removeBuddy()`, and `getBuddies()`.
- Person**: A base class for **Buddy** and **Contact**.
- Buddy**: Contains `getName()` and `getBuddies()`.
- Contact**: Contains `getName()` and `getBuddies()`.
- Friend**: A subclass of **Contact**.
- Friendship**: A class representing relationships between buddies.

11 Feb      8

University of Twente  
The Netherlands

### Approach

This diagram is identical to slide 7, showing the flow from service specification to platform selection.

11 February 2009      TTT      9

University of Twente  
The Netherlands

### Example: Service Design Refined Model(1)

#### Live Contacts Architecture

The Live Contacts Architecture diagram shows the interaction between various components:
 

- User** and **Buddy** interact with **Presentation Component** and **User Agent**.
- User Agent** sends **user input events** to the **Service Coordinator**.
- Context Sources** (GPS, MSN, Outlook) provide **context events** to the **Service Coordinator**.
- The **Service Coordinator** manages a **Database** and interacts with **Action Providers** (SMS, Phone, Email, Chat services).
- Service Trader** handles **register** and **discover** operations.

11 February 2009      TTT      10

University of Twente  
The Netherlands

### Example: Service Design Refined Model(2)

This diagram shows a service design refined model with numerous classes and their interactions. Key elements include:
 

- ContextSources**: A collection of context sources.
- ServiceCoordinator**: The central component managing the system.
- Database**: Used for storing and retrieving data.
- ActionProviders**: Implement specific actions like SMS, Phone, Email, and Chat.
- ServiceTrader**: Manages the discovery and registration of services.

11      11

University of Twente  
The Netherlands

### Example: Service Design Refined Model(3)

**Interaction patterns**  
'Recurring sequence of actions performed by two or more interacting components'

**Basic interaction patterns** occur between two interacting components

**Composite interaction patterns** occur between more than two components (combinations of basic patterns)

11 February 2009      TTT      12

University of Twente  
The Netherlands

### Approach

11 February 2009      TTT      13

University of Twente  
The Netherlands

### Example: Service Design Component Model

11 February 2009      TTT      14

University of Twente  
The Netherlands

### Transformation $T_1'$ with Medini QVT

**Source metamodel:**  
A-MUSE DSL metamodel  
**Target metamodel:**  
A-MUSE DSL metamodel

**Source model:**  
A-MUSE DSL service specification  
**Target model:**  
A-MUSE DSL service design refined model

11 February 2009      TTT      15

University of Twente  
The Netherlands

11 February 2009      TTT      16

University of Twente  
The Netherlands

```

transformation ServiceSpecificationToServiceDesignRefinedModel (ss: asdl, sdrm: asdl)
{
  top relation BehaviourMapping {
    checkonly domain ss ssBehaviour: Behaviour {name = 'ServiceSpecification'};
    enforce domain sdrm sdrmBehaviour: Behaviour {name = 'ServiceDesignRefinedModel'};
  }

  top relation EntryPointMapping {
    entryPointName: String;
    checkonly domain ss ssEntryPoint: Entry {
      name = entryPointName,
      parent = ssBehaviour: Behaviour{}
    };
    enforce domain sdrm sdrmEntryPoint: Entry {
      name = entryPointName,
      parent = sdrmBehaviour: Behaviour{}
    };
    when {BehaviourMapping(ssBehaviour,sdrmBehaviour);}
  }
}

```

11 February 2009      TTT      17

University of Twente  
The Netherlands

### Conclusions

**Our MDA-based approach divides the PIM level in**

- ▶ tree models:
  1. Service specification
  2. Service design refined model
  3. Service design component model
- ▶ Two transformations
  1. Service specification -> Service design refined model
  2. Service design refined model -> Service design component model

**Towards the automation of this approach, we have realised the transformation  $T_1'$  with the Medini QVT tool**

11 February 2009      TTT      18

## Future work

### Further study on transformation $T_1'$

- ▶ Granularity of interaction patterns: trade-off between automation and flexibility
- ▶ Extension of transformation rules to cover the Live Contacts application
- ▶ Testing of these extended transformation rules with other context-aware mobile applications

### Automation of transformation $T_2'$

- ▶ Investigation of formalisms to support behaviour synthesis, coping with synchronization and concurrency issues in the behaviour of interacting components