

# Model Driven Integration of Software Systems

Alek Radjenovic

HISE Seminar  
18 Oct 2011

2

HISE Seminar, 18 Oct 2011

## Outline

- Context & Problems
- Key objectives
- Solution
  - Concepts
  - Implementation
- Case studies

3

HISE Seminar, 18 Oct 2011

## Why model?

- We've always modelled
  - simplify the reality
    - better understand the world around us
    - design and develop objects/systems in a gradual/incremental fashion using structured approach
    - apportion various aspects of design to the 'experts' (a.k.a. domain specialists)
      - used both in development as well as analysis
  - Humans use abstraction to better deal with complexity
    - by excluding the unnecessary detail
  - Abstraction (of a system, or its environment, or both)
    - is the key tool in modelling

4

HISE Seminar, 18 Oct 2011

## Software modelling: a historical perspective

- 50's and 60's – abstraction of 0's and 1's into:
  - symbolic names, high-level operations, block structures, ...
- 70's – system design
  - functional isolation, API, graphical representation, ...
- 80's – OO, software architecture
- 90's – MDE/MBDE
  - UML, ADLs, DSLs, ...

5

HISE Seminar, 18 Oct 2011

## Software modelling: a modern view

- (almost) all software artefacts are models
  - requirements specifications
  - design (software architecture) specifications
    - e.g. UML diagrams
  - source code
  - database schemas
  - petri nets
  - even a directory listing from your hard disk!

6

HISE Seminar, 18 Oct 2011

## Context & Problems

- Large scale, complex, software → Distributed teams → Heterogeneous system components (legacy and new) → Late, sometimes unpredictable, costly system integration
- Increased use of model driven engineering + Multiple (standard and non-standard) modelling platforms + Varying versions of modelling tools and languages = Inability to integrate easily at model level
- Integration at modelling stage → Early detection of incompatibilities → (potentially huge) cost saving during system integration

## Ultimate objective: Model Integration

- Questions to answer:
  - Is model integration possible?
  - To what extent?
  - How to deal with identified incompatibilities?
    - Automatic? Manual? Guidance only?
- Beyond scope
  - Model maintenance; Model evolution
- Blue-skies research for SSEI
  - not 'without a clear goal', but certainly 'curiosity-driven science'

## State-of-the-art

- Model management techniques
  - Model compatibility
    - initiative for a unifying model (e.g. MOF)
      - theoretically – a step in the right direction
      - in practice – lack of widespread support by tools; versions
  - Model comparison and differencing
    - first 2 steps in assessing incompatibilities
    - SiDiff – graphical representations (UML, Simulink)
    - EMF Compare – Eclipse (EMF input models)

## State-of-the-art (2)

- Model Transformation
  - ATL – Eclipse based; robust tool support
    - language somewhat cumbersome
    - substantially declarative nature (e.g. iterations over complex structures not easy)
  - ETL – Eclipse based (Epsilon)
    - hybrid (declarative & imperative)
      - imperative part: loops, assignments, statement sequencing
    - reuses a portion of OCL for navigating model elements
  - VIATRA2 – UML; graph transformations & ASMs

## State-of-the-art (3)

- Model Composition (Merging)
  - AMW (ATLAS Model Weaver) – Eclipse based
  - EML (an Eclipse/Epsilon language)
  - ReuseWare – for Semantic Web (OWL, Xcerpt, XQuery)
  - openArchitectureWare – Eclipse based
  - Kermeta – meta-programming environment based on MOF (Eclipse based tool)
- Conclusion
  - no integrated approaches
  - key issue: different meta-models (languages)

11

HISE Seminar, 18 Oct 2011

## Conceptual approach

- Decomposition of models
  - Structural (mostly evident at syntax level)
  - Behavioural (semantics)
- Compatibility checking
  - Structural: static; production of correspondence models
  - Behavioural: dynamic; simulation + (post-mortem) analyses

12

HISE Seminar, 18 Oct 2011

## Solution in a nutshell

- Input models
  - Heterogeneous (theoretically any modelling platform)
  - Textual representation
- Interchange platform
- Compatibility checking
  - Structural
  - Behavioural
- Model integration
- Proof of the pudding: large case study (avionics)

13

HISE Seminar, 18 Oct 2011

## Solution platform

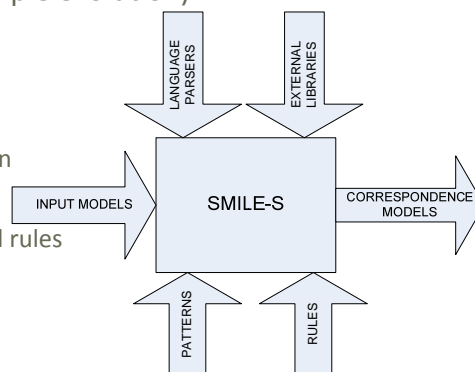
- SMILE (Simple Model Integration Languages with Execution engines)
- Three key components (languages + tools):
  - SMILE-S (model structure)
  - SMILE-X (model behaviour)
  - SMILE-I (model integration)
- Key objectives
  - 'meta-model agnostic'
  - extensible
  - scalable

14

HISE Seminar, 18 Oct 2011

## SMILE-S as a black box

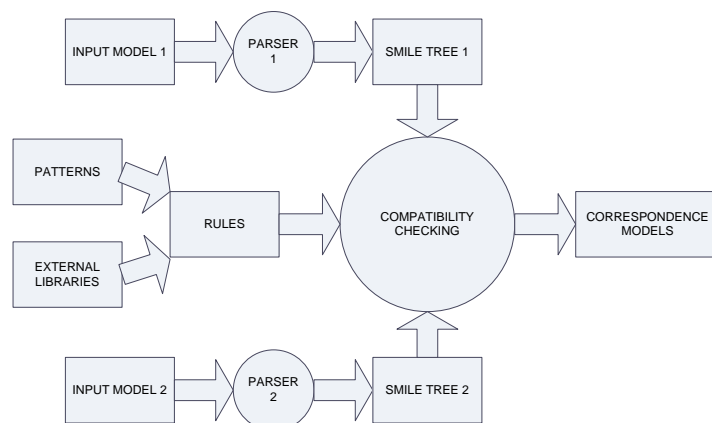
- Philosophy: minimal core, extensible
- External to the core (simple evolution):
  - Pattern language
  - Rule language
  - Language parsers
    - one way transformation
  - Libraries of routines
    - for use by patterns and rules
- 2 input models only



15

HISE Seminar, 18 Oct 2011

## SMILE-S: Internal architecture



16

HISE Seminar, 18 Oct 2011

## SMILE-S: Interchange format

- tree-based
  - nodes = model elements, links = containment relationship
- attributes: name, native type, native ID, SMILE type, SMILE ID
- properties: dependent on the model

node
<b>ATTRIBUTES</b> A[KEY] = VALUE
<b>PROPERTIES (optional)</b> P[KEY] = (VALUE, TYPE)
<b>CHILDREN (optional)</b> C1, C2,...,CN: node

17 HISE Seminar, 18 Oct 2011

## SMILE-S trees

**Left model**

```

softarch
├── database (softarch)
│   ├── table (auth_notes)
│   │   └── row (0)
│   │       ├── column (ID)
│   │       ├── column (AuthorID)
│   │       └── column (Date)
│   ├── table (auth_paper)
│   ├── table (conference)
│   ├── table (identity)
│   ├── table (issue)
│   ├── table (journal)
│   └── table (papers)
└── Properties
    ├── type: datetime
    └── value: 04/12/2003 23:16:30
        
```

**Right model**

```

Lift
├── xmi:XML
│   ├── xmi:Documentation
│   └── uml:Model (EA_Model)
│       ├── packagedElement (Lift)
│       ├── packagedElement (EA_Collaboration1)
│       ├── packagedElement (Passenger)
│       ├── packagedElement (Button)
│       └── packagedElement (Controller)
│           ├── packagedElement
│           ├── packagedElement
│           ├── packagedElement (Door)
│           ├── ownedAttribute (Closed)
│           │   ├── lowerValue
│           │   └── upperValue
│           └── type
│               ├── ownedOperation (close)
│               ├── ownedOperation (open)
│               └── ownedAttribute
│                   └── packagedElement
│                       └── packagedElement (Floor)
        
```

**Properties**

name	Controller
visibility	public
xmi.id	EAID_7E7E6FE5_AF9D_4914_8F48_BAC437BD2AA9
xmi.type	uml:Class

18 HISE Seminar, 18 Oct 2011

## SMILE-S: Plug-ins/Extensions

- (language/model) Parsers
  - Meta-model knowledge displaced externally
- Comparison/Matching/Compatibility routines
  - typically return Boolean values

**Parser Manager**

Name	Files	Assembly	Version	Publisher
Dir	*.dir	D:\York\SSEI\DEV\SMILE\SMILE-S\...	1.0.0.1	SSEI
SQL	*.sql	D:\York\SSEI\DEV\SMILE\SMILE-S\...	1.0.0.1	SSEI

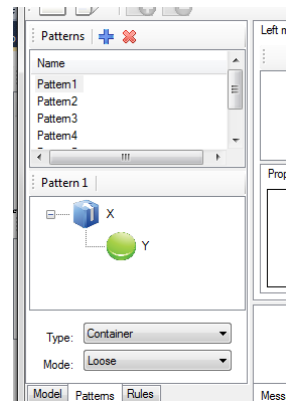
**Smile-S (D:\York\SSEI\DEV\Projects\Smile-S)**

File Plug-ins Model Patterns Rule

Parsers...  
External libraries...

## SMILE-S: Patterns

- Specify constellations of
  - 1..\* nodes
  - 0..\* 'filters' (match criteria)
    - Y.Property[xmi:type] = uml:class
- Applied to a single model



## SMILE-S: Rules

- Specification
  - left and right pattern ↔ correspond to input models
  - (a set of) matching conditions
    - Boolean logic + internal/external routines
- Application
  - returns a pair of compatible nodes/subtrees (or NULL)
  - has attached
    - a similarity score
    - trace

```

<rule name="abc">
  <pattern>
    <left>
      <X type="container" mode="loose">
        <Y type="leaf"/>
      </X>
    </left>
    <right>
      <Z type="container"/>
    </right>
  </pattern>

  <match>
    <left value="Y[value]"/>
    <right value="Z[name]"/>
    <method value="SMILE.Compare.IgnoreCase"/>
  </match>
</rule>

```

21

HISE Seminar, 18 Oct 2011

## SMILE-S: Correspondence models

The screenshot shows two windows side-by-side. The left window, titled 'pubs.dos', displays a directory tree with a yellow header and a green background. The right window, titled 'cognitat.sql', displays a database schema tree with a yellow header and a green background. A green callout box on the left side of the 'pubs.dos' window contains the following text:

- Colour coded trees
- Provide visual guidance to incompatibilities
- Contextual information details failed rules

22

HISE Seminar, 18 Oct 2011

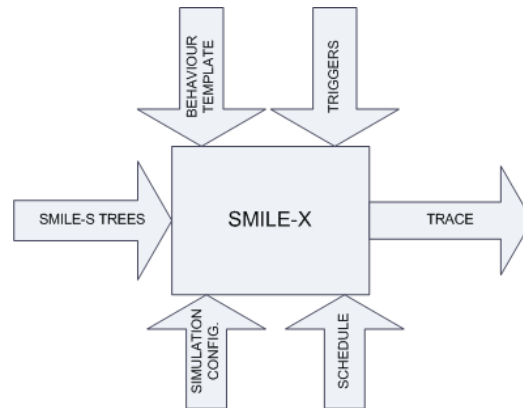
## SMILE-S: Summary

- Interchange format (trees)
- Pattern language
- Rule language
- Execution engine
- Correspondence models
- Plug-ins
  - parsers:
    - XMI (UML, SysML), Java, (modified) SQL, a DSL (DOS directory listing)
  - comparison libraries

23

HISE Seminar, 18 Oct 2011

## SMILE-X as a black box

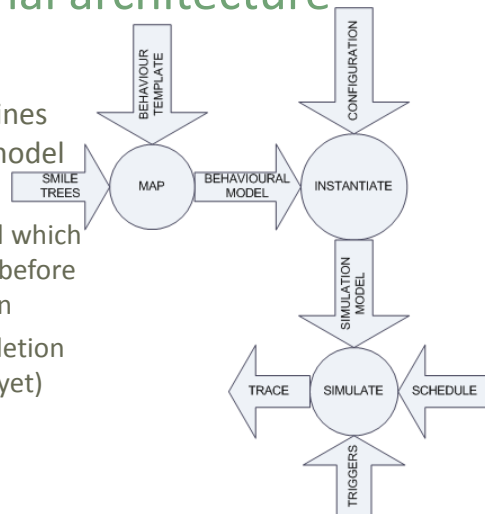


24

HISE Seminar, 18 Oct 2011

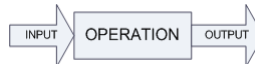
## SMILE-X: Internal architecture

- Behavioural model combines template artefacts with model elements
- effectively, a meta-model which needs to be instantiated before we can run the simulation
- dynamic creation and deletion of objects not a feature (yet)



## SMILE-X: Behaviour

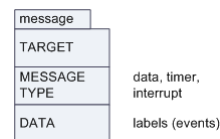
- Basic unit of behaviour



- Messages (input or output)

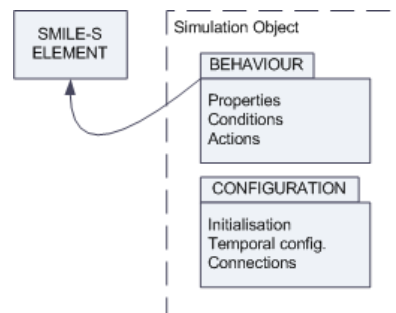
- Operations

- have duration (e.g. WCET)
- collection of actions
- can have condition (just a Boolean label - evaluates to both true and false during simulation)
  - can be linked with action



## SMILE-X: Simulation objects

- SMILE-S tree elements + Behavioural template = Behavioural Model
  - a collection of behavioural elements
- Behavioural model + Configuration = Simulation Model
  - a collection of simulation objects



27

HISE Seminar, 18 Oct 2011

## SMILE-X: Triggers

- a (compound) Boolean expression
- Flag to stop execution or continue
- Example:

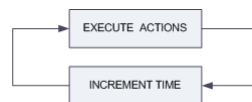
Trigger_LiftBusy			
AND OR			
	Object	Operator	Value
	Controller.State	==	ACTIVE
	Controller.Input	==	LIFT_REQUEST
	Controller.Output	==	MOVE
**			

28

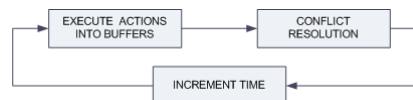
HISE Seminar, 18 Oct 2011

## SMILE-X: Schedules

- Simple activation



- Double buffer



- Event based

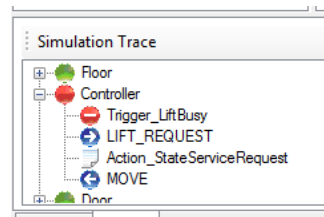


29

HISE Seminar, 18 Oct 2011

## SMILE-X: Traces

- Sequential
- Provide information on:
  - input message
  - failed conditions
  - executed actions
  - output message
  - triggered conditions
- Colour coded



30

HISE Seminar, 18 Oct 2011

## SMILE-X: Summary

- Uses SMILE-S models (trees)
- Adds semantics to nodes and properties
- Uses a template to create behavioural model:
  - maps between behavioural concepts and structural elements
  - specifies allowed interactions
- Manual instantiation of behavioural model elements to create simulation objects
- Trigger specification (manual)
- Schedule selection
- Trace inspection

31

HISE Seminar, 18 Oct 2011

## SMILE-I: Integration component

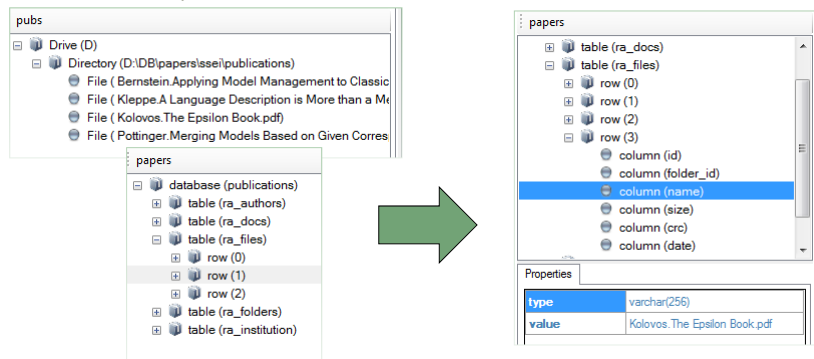
- Small, in-house case studies
  - File system (DOSDir DSL) + Documents database (SQL)
  - Lift system (UML interaction diagrams)
- Model integration
  - based on model weaving
  - examples very basic
    - need real-world case studies

32

HISE Seminar, 18 Oct 2011

## SMILE-I: Example

- Before: 4 physical files, 3 database records (rows)
- After: 4 files, 4 rows



33

HISE Seminar, 18 Oct 2011

## SMILE-I: Model weaving

- Requires additional plug-in for each language
- Key operations (parameters):
  - creation, deletion (node/subtree)
    - based on unmatched patterns
  - insertion (insertion point/parent node)
  - move (node/subtree, new parent)
    - manual intervention (at the moment)
    - future: rule-based

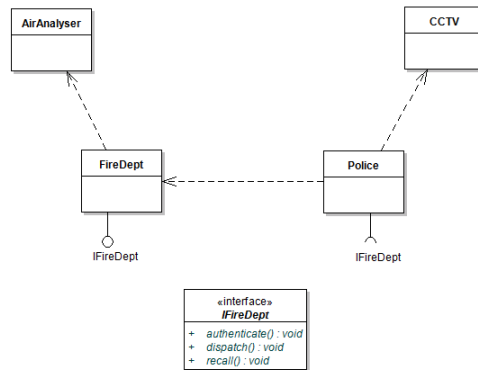
34

HISE Seminar, 18 Oct 2011

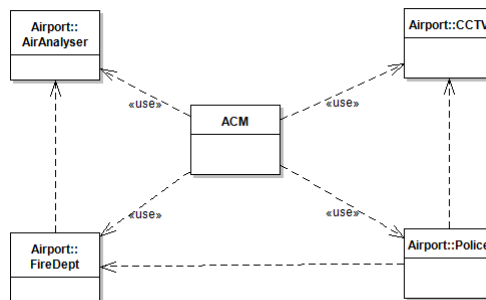
## Tools: Implementation summary

- GUI driven
- Microsoft Windows
- C# and .NET
- Extensible via plug-ins
- Project based

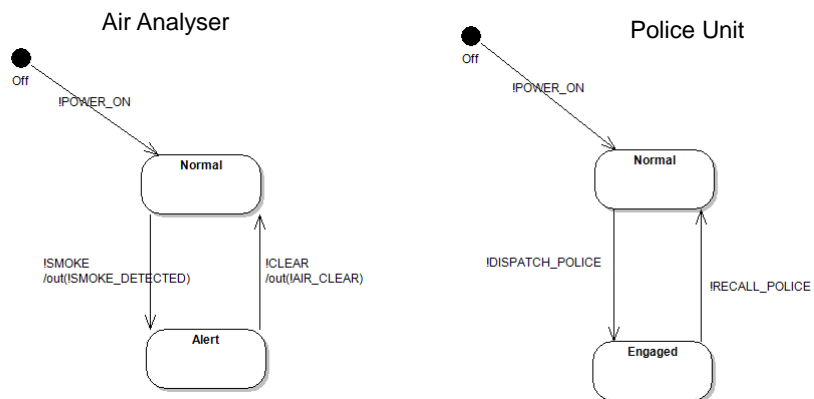
## Case study: Airport (legacy)



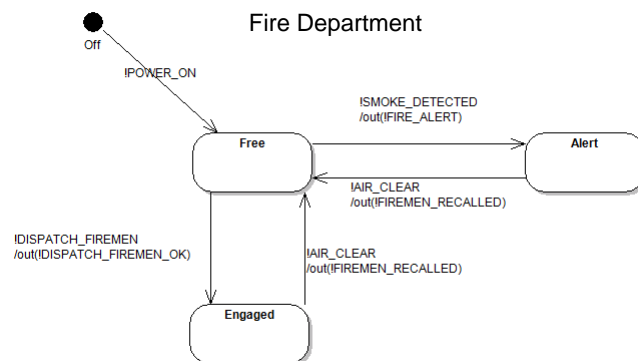
## Case study: Airport (new)



## Case study: behaviours

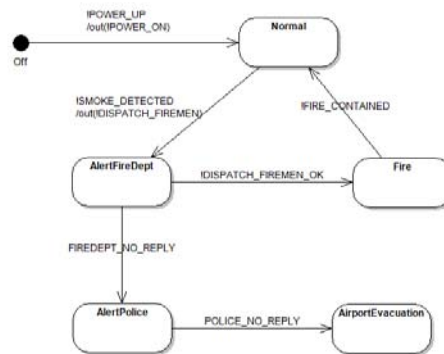


## Case study: behaviours



## Case study: behaviours

ACM (Airport Crisis Management unit)



## Refereed papers

- Radjenovic, Paige, "An Approach for Model Querying-by-Example Applied to Multi-Paradigm Models", 5th Intl. Workshop on Multi-Paradigm Modeling (MPM 2011), MODELS 2011 (ECEASST)
- Radjenovic, Paige, "Behavioural Interoperability to Support Model-Driven Systems Integration", 1st Intl. Workshop on Model-Driven Interoperability (MDI 2010), MODELS 2010 (ACM)

## Reports

- [SSEI-TR-0000015] "**State-of-the-art Survey of Model Driven Integration of Software Systems**"
- [SSEI-TR-0000042] "**Structural Compatibility in Models**"
- [SSEI-TR-0000043] "**Behavioural Compatibility in Models**"
- [SSEI-TR-0000095] "**Model Driven Integration: CMS-1 CS Case Study**"

## Links

- MOF - <http://www.omg.org/mof/>
- SiDiff - <http://sidiff.org/>
- EMF Compare - <http://www.eclipse.org/emf/compare/>
- ATL – <http://www.eclipse.org/atl>
- ETL, EML - <http://www.eclipse.org/gmt/epsilon/>
- VIATRA - <http://www.eclipse.org/gmt/VIATRA2/>
- AMW - <http://www.eclipse.org/gmt/amw/>
- ReuseWare - <http://www.reuseware.org/>
- openArchitectureWare - <http://www.openarchitectureware.org/>
- Kermeta - <http://www.kermeta.org/>