Towards V&V suitable Domain Specific Modeling Languages for MBSE

Blazo Nastov\textsuperscript{1}, Vincent Chapurlat\textsuperscript{1}, Christophe Dony\textsuperscript{2} and François Pfister\textsuperscript{1}

\textsuperscript{1}LGI2P, Ecole des mines d’Alès, Nîmes, France
\{blazo.nastov, vincent.chapurlat, francois.pfister\}@mines-ales.fr
\textsuperscript{2}LIRMM, University of Montpellier, Montpellier, France
dony@lirmm.fr

Modelling Complex Systems

Model Based Systems Engineering (MBSE)
Approaches for systems modelling
Based on creating, manipulating and analyzing models

Mainly, a model is...
A partial and incomplete, may be false but crucial representation of a system
An argument for decision making processes
Problematic and Needs

1) To create Domain Specific Modeling Language (DSML) suitable for modelling a system (multi disciplinary, multi views / aspects, ...)

A system of interest
**Problematic and Needs**

1) To create Domain Specific Modeling Language (DSML) suitable for modelling a system (multi disciplinary, multi views / aspects, ...)

2) Model V&V: assure model’s well-formedness, coherence and conformity (Verification), justifiability and relevance for expert’s objectives (Validation)

**Agenda**

**Background**

*Creating DSML: things to consider*

*V&V strategies*

**Contributions**

*Conceptual*

*Methodological*

*Technical*

**Case study**

**Conclusion and Perspectives**
Creating a DSML: things to consider

Syntax
   Abstract syntax
   Concrete syntax

Semantics
   Dynamic semantics (behavior):
      Translational semantics
      Operational semantics
         Action languages (Kermeta, Java,...) => Operations, methods, functions,
         Behavioral modeling languages (Statechart, State machine,...) => behavioral models

Static semantics (Formal properties):
   Structural properties (temporal or not)
   Behavioral properties (temporal or not)
V&V strategies

DSMLs lack semantics: Guided modeling and Experts model evaluation

DSMLs include semantics: Model simulation and Formal verification of properties

Based on 3rd party approaches (M2M transformation)

(+): Reuse of existing approaches

(-): Information loss, relevance between the source and the target

“Direct” simulation and property verification

(+): Behavior and properties are directly defined on concepts, Simulation and property verification as early as possible

V&V strategies

DSMLs lack semantics: Guided modeling and Experts model evaluation

DSMLs include semantics: Model simulation and Formal verification of properties

Based on 3rd party approaches (M2M transformation)

(+): Reuse of existing approaches

(-): Information loss, relevance between the source and the target

“Direct” simulation and property verification

(+): Behavior and properties are directly defined on concepts, Simulation and property verification as early as possible
An approach for designing V&V suitable DSMLs

Contribution: a conceptual and tool-equipped approach for the design of DSMLs allowing simulation and formal verification of properties
Conceptual contributions: overview

**eXecutable, Verifiable and Interoperable Core: xviCore**

Promotes a formalized modeling lifecycle process based on several formalized phases and sub-phases

| xviCore: Meta language composed of three languages |
|-----------------------------------------------|---------------------------------|--------------------------------|
| Object-Oriented metamodeling language         | Property modeling language     | Formal behavioral modeling language |

---

**Conceptual contributions: overview**

**Phase 1: DSML design time**

---

**Language properties**

- Structural properties
- Behavior properties

**Operational semantics specification**

---

```
→ « instanceOf » ← « basedOn » ← « executes »
```
Conceptual contributions: overview

**eXecutable, Verifiable and Interoperable Core: xviCore**

**Phase I: DSML design time**

**Phase II: DSML run time**

**Phase IIA: Model design time**

**Phase IIB: Model run time**

**xviCore: Meta language composed of three languages**

- **Object-Oriented metamodeling language**
- **Property modeling language**
- **Formal behavioral modeling language**

**DSML**

- **Abstract syntax (metamodel)**
- **Language properties**
  - Structural properties
  - Behavioral properties
- **Operational semantics specification**

**Model**

- **Model abstract syntax**
- **Model and Object properties**

**Type**

- Discrete
- Continuous
- Hybrid

**Simulation**

- `instanceOf` 
- `basedOn` 
- `executes`
**Methodological contribution**: simulation mechanisms

**Observations**
Simulation or model execution requires the specification of behavior
- Multiple evolving concepts in a DSML
- Even more if considering multiple aspects conceptualized by multiple DSMLs

**Problem**: coordinate all behavioral models from one or several DSMLs for simulation

**State of the art**: The blackboard design pattern [Engelmore and Morgan, 1988]

![Blackboard diagram](image)


---

**Objective**: applying the blackboard design pattern in the MBSE context

![Blackboard diagram](image)

**Methodological contribution: simulation mechanisms**

**Objective:** applying the blackboard design pattern in the MBSE context

- Schedules and executes behavioral models
- Based on a simple execution algorithm
  - Read inputs from blackboard
  - Calculate future state
  - Write outputs into blackboard
- and other original rules

---


---

**Methodological contribution: simulation mechanisms**

**Objective:** applying the blackboard design pattern in the MBSE context

- Allows data exchange between behavioral models

---

Methodological contribution: simulation mechanisms

Objective: applying the blackboard design pattern in the MBSE context

Proof of concept: interpreted sequential machine (ISM)

Advantages of ISM [Vandermeulen, 1996]
- Operates with typed data
- Separates state/transition from data specification
- Underlying structure based on the LTL (Linear Temporal Logic)


Methodological contribution: verification mechanism

A verification process requires:
- Formal specification (DSML syntax, DSML dynamic semantics, model)
- Formal properties (need for a property modeling language)
- A model-checking tool (need to reuse existent or create a new tool)

Verification activities proposed by the xviCore approach
- Well-formedness of the structure
  - DSML and Model structural properties
- Well-formedness of the behavior
  - DSML behavioral properties

Used property modeling language: CREI [Vincent, 2014]

Reuse of existent tools
- Structural properties: rewriting CREI properties to OCL
- Behavioral properties: rewriting CREI properties to OCL
Technical contribution: current editors

DSML design
Syntax

[Diagram]

Technical contribution: current editors

DSML design
Syntax
Behavior

[Diagram]
Technical contribution: current editors

DSML design
  Syntax
  Behavior
  Properties

Model design

Resource - Sample/tec/tecoeditor - Eclipse Platform

Property P1
  Modality always possible
  Case [forall f in function / exists a in flow]
  EFFECT [f.ArrayAdapter]<

Resource Set
  platforms/resource/test/1.toto
    Systeme Devu
    Resource 2.3
    Vannes 0.0
    Vannes 0.0
    Controller
Technical contribution: extending EMF

We have extended the code generation library of EMF

Generation of a simulation library
Technical contribution: extending EMF

We have extended the code generation library of EMF

Generation of a simulation library

Simulation trace

Case study based on two DSMLs: EFFBD and PBD
Example: enhanced Functional Flows Block Diagram (eFFBD)

Functional / Behavioural model (dynamics of a system)

Example: enhanced Functional Flows Block Diagram (eFFBD)

Function

Functional / Behavioural model (dynamics of a system)
Example: enhanced Functional Flows Block Diagram (eFFBD)

Flow (I/O, Trigger)

Functional / Behavioural model (dynamics of a system)

Example: enhanced Functional Flows Block Diagram (eFFBD)

Construct (parallelism, choice, loop, ...)

Functional / Behavioural model (dynamics of a system)
Example: enhanced Functional Flows Block Diagram (eFFBD)

Request: to define a behavior
Example: enhanced Functional Flows Block Diagram (eFFBD)

**EFFBD abstract syntax**

**EFFBD model**

**Example:** to define a behavior

**EFFBD dynamic semantics (for Function)**

- **PC**
  - **update:** transformingInputs
  - **condition:** itemInputs.all(present) AND
    - **Execution**
      - **Authorized**
      - **Execution**
        - **updated:** transformingInputs
        - **events:** suspend
        - **condition:** internalTime-startedTime > 2
        - **Finished:** outputItem.all(present)
        - **Sleep**
        - **Aborted**

Example: Physical Block Diagram (PBD)

Model of architectural composition (the physical components of a system)
Example: Physical Block Diagram (PBD)

Model of architectural composition (the physical components of a system)
Example: Physical Block Diagram (PBD)

Model of architectural composition (the physical components of a system)

Example: Physical Block Diagram (PBD)

PBD abstract syntax

PBD model
Example: Physical Block Diagram (PBD)

PBD abstract syntax

PBD model

Request: to define a behavior

Example: Physical Block Diagram (PBD)

PBD abstract syntax

PBD model

PBD dynamic semantics (for Component)

Request: to define a behavior

Event: start

Event: internalBreakdown

Event: resume

Event: externalBreakdown

Event: deactivate

Event: update

Event: notify
Dependencies between eFFBD and PBD

“physical components of a system perform one or more functions, in the same ways, functions are allocated to a component”

EFFBD model

PBD model

Dependencies between eFFBD and PBD

“physical components of a system perform one or more functions, in the same ways, functions are allocated to a component”

EFFBD model

PBD model
Dependencies between eFFBD and PBD

“physical components of a system perform one or more functions, in the same ways, functions are allocated to a component”

1) Structural Dependencies
2) Behavioral Dependencies

Phase I: DSML design time and DSML property specification
Phase I: DSML design time and DSML property specification

**Structural dependencies**

**EFFBD DSML**

- Function (1..1) message (0..*) function (1..1) performs
- Context (0..1) buildFrom

**PBD DSML**

- Component (0..1) source (0..1) target
- Link (0..1) externalSource (0..1) externalTarget

**Behavioral dependencies**

- notify: Outputs.setFunction, state
**Phase I: DSML design time and DSML property specification**

DSML “structural” property: The quantity of a Resource that a function provides or requires for execution must be positive or zero is specified as

\[ C := \forall f \in \text{Function} \mid \forall r \in f. \text{resourceFlowInput} | r. \text{requestedQuantity} \geq 0 \]

\[ R := \forall f \in \text{Function} \mid \forall r \in f. \text{providings} | r. \text{providedQuantity} \geq 0 \]

DSML “behavioral” property: If a component enters a breakdown state (internal or external), its functions will be unable to continue execution

\[ C := \forall c \in \text{Component} \mid c. \text{State} = \text{SS} \text{ OR } c. \text{State} = \text{ES} \]

\[ R := \forall f \in \text{Function} \mid f. \text{State} \neq \text{Execution} \]

**Phase II.A: DSML run time | Model design time and Model property specification**

Design model structural properties: If the AI unit is alerted of an ongoing threat, it must send a report to the surveillance center, even if this threat appears not to be an incident

\[ C := \text{AIUnit.mission.getInputs("Fire detected")} \neq \emptyset \text{ OR } \text{AIUnit.mission.getInputs("Flood detected")} \neq \emptyset \]

\[ R := \Rightarrow \]

\[ E := \text{SurvCent.mission.getInputs("Situation report")} \neq \emptyset \]
Phase II.B: DSML run time | Model run time

Detecting Fire
  Execution Suspended
Fire Detector
  Producing External Stop

Global time scale

Phase II.B: DSML run time | Model run time

Detecting Fire
  Execution Suspended
Fire Detector
  Producing External Stop

Global time scale
Phase II.B: DSML run time | Model run time

Detecting Fire
Execution Suspended
Fire Detector
Producing External Stop

Global time scale

Phase II.B: DSML run time | Model run time

Detecting Fire
Execution Suspended
Fire Detector
Producing External Stop

Global time scale
**Phase II.B: DSML run time | Model run time**

```
Detecting Fire
  Execution Suspended
Fire Detector Producing External Stop

Global time scale
  x  x+1  x+2  x+3  x+4  x+5
  Restart Production
```

```
Detecting Fire
  Execution Suspended
Fire Detector Producing External Stop

Global time scale
  x  x+1  x+2  x+3  x+4  x+5
  Restart Production
```
**Phase II.B: DSML run time | Model run time**

![Diagram showing the flow of events and timelines for Detecting Fire, Execution Suspended, Fire Detector, Producing External Stop.](image)

**Conclusion and Perspectives**

**Research focus**
Modelling Complex Systems

**Contributions**
An approach for the design of DSMLs and models
Simulation mechanism
Specification of properties and their verification

**Tool support**
Tool for the simultaneous design of DSML’s syntax, behavior and property specification
Modification of EMF metamodeling editor and code generator
Tool for simulation and property verification

**Case study**
eFFBD and PBD

**Future works**
Continuous and Hybrid behavioral models
Papers 2014/2015

2014


2015


B. Nastov, V. Chapurlat, C. Dony and F. Pfister. "Model Verification & Validation and Model Based Systems Engineering: towards executable DSML”. INSOCE Magazine INSIGHT

Papers 2016

2016


Thank you for your attention

Questions?