

Context-aware Model Driven Development by Parameterized Transformation

Samyr Vale, Slimane Hammoudi

Sponsor:
FAPEMA (Brazil)

ESEO – Angers – France
LERIA – Université d'Angers

Research Areas/ Motivation

Research Domains:

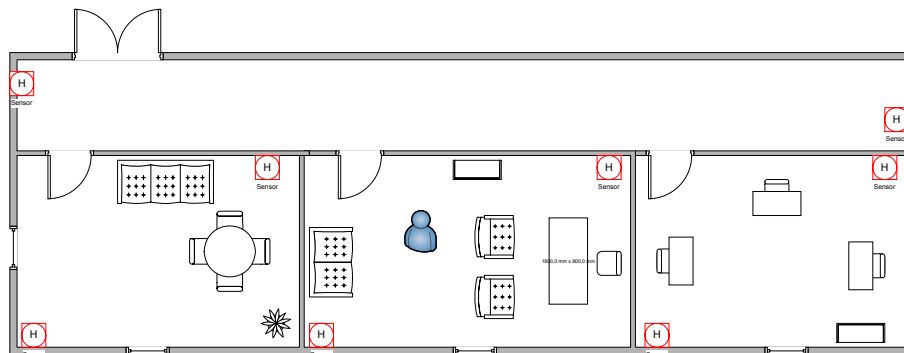
- Ubiquitous/Pervasive Computing
- Model Driven Development
- Ontology
- Web Services

Motivation:

- Problems
 - Context (data and activities) into Business Logic
 - Context Dependence
 - No-pattern / No-approach
 - Legacy artifacts and ad hoc architectures
 - Reuse, Scalability, Interoperability = fastidious task

- ❑ Context-aware and Context-awareness,
- ❑ Problems,
- ❑ Model Driven Approach,
- ❑ Proposed solution by the MDE approach,
- ❑ Interoperability, Scalability, Reuse...
- ❑ Development by individual Models and different Abstraction Levels,
- ❑ Parameterized Transformation Process
- ❑ Match and Adaptation

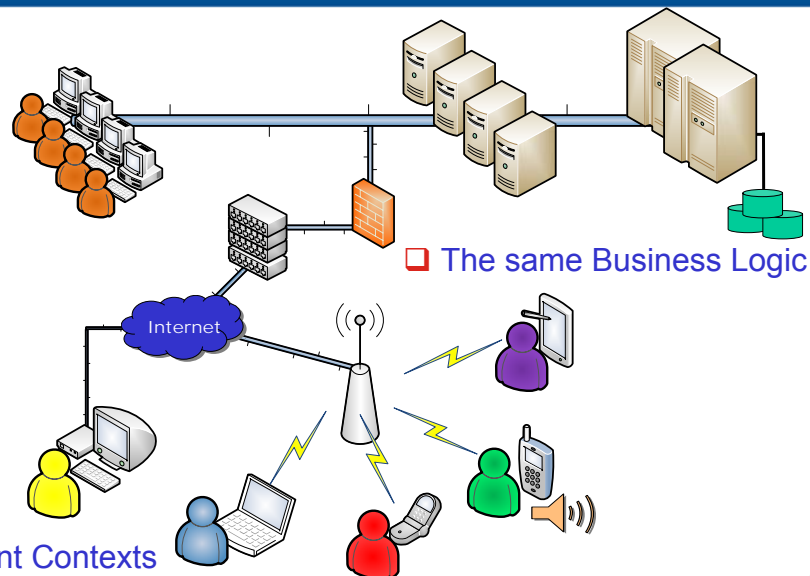
- ❑ Sensors Use
- ❑ Localization
- ❑ Quotidian Activities



Context-aware and Context-awareness (Dey's Vision, 1999)

- ❑ Context is everything that characterize an entity.
- ❑ An entity is any element involved in the computational environment.
- ❑ E.g. Person, Local, Time, Device, Network, Profile . . .
- ❑ Complexity.
- ❑ Legacy artifacts and ad hoc architectures.
- ❑ ~~Pattern and Methodology.~~
- ❑ ~~Reuse, scalability, interoperability.~~

Context-aware and Context-awareness (Dey's Vision, 1999)



Context-aware and Context-awareness

User		
Usage		
Element	Total	Description
1	1	John
1	1	Laptop

- Context data
- Business data
- Context-aware activities
- Business activities

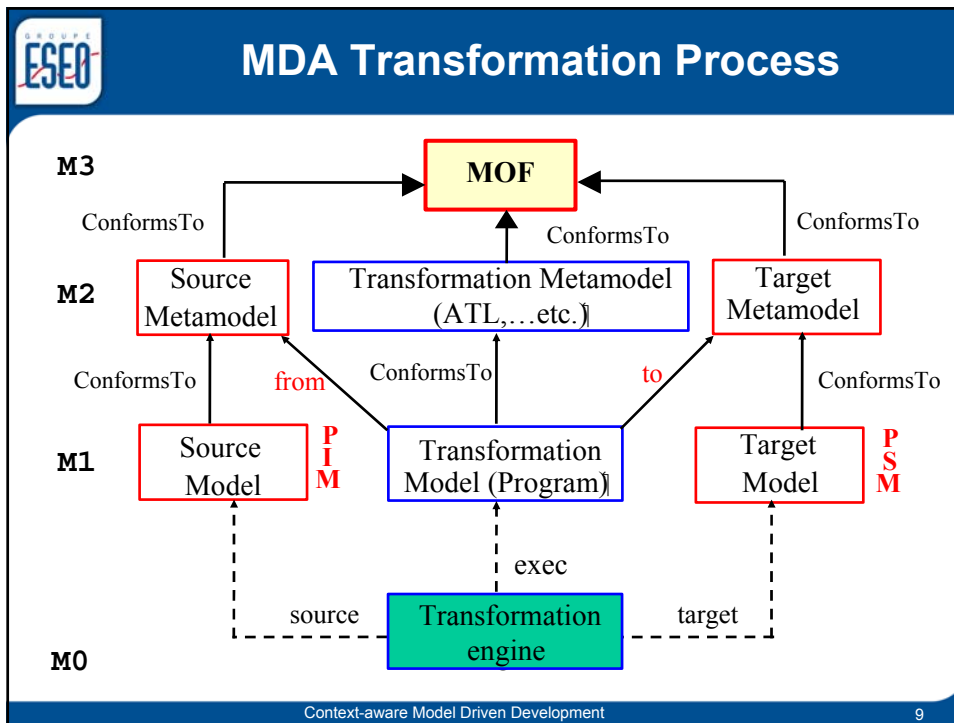
- Profile
- Device Type
- Time
- Location
- Interface
- Network

Context-aware Model Driven Development 7

Model Driven Engineering

- MDA (Model Driven Architecture) = OMG`s standard for MDE
- PIM and PSM models development
- Different abstraction levels
- Mapping and Transformations Techniques
- From Meta-metamodel Level until Code Level
- Different programming languages, operational systems, platforms, data bases.
- Reuse, scalability, Interoperability.

Context-aware Model Driven Development 8



Context and MDA

Some challenges:

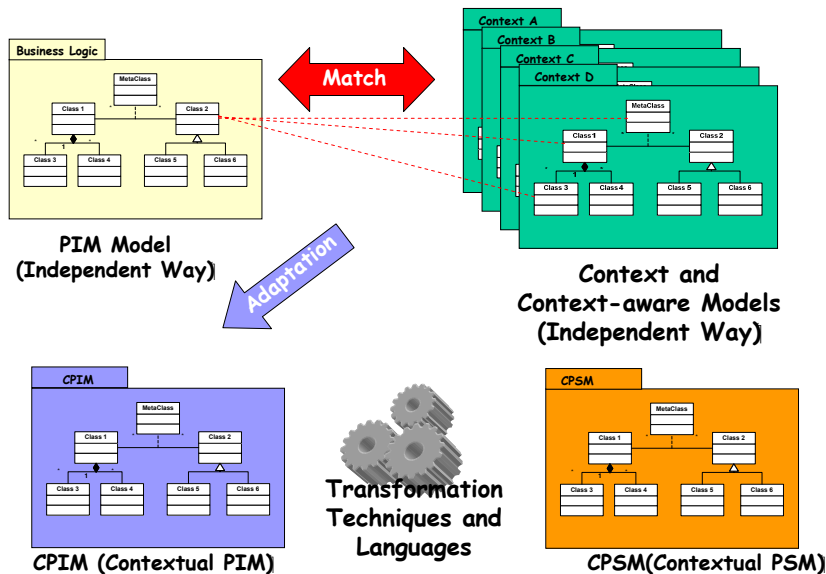
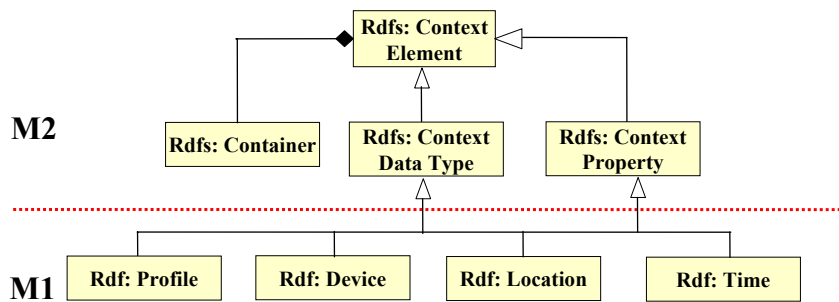
- Context Capture
- Context Representation
- Context Interpretation
- Context Adaptation
- Context Management
- Context Reuse

Some solutions:

- Interpreters
- Wrappers
- Aggregators
- Parsers
- Adaptors
- Legacy architectures and others artifacts

Context-aware Model Driven Development 10

- ❑ Context Metamodel (OMG's Ontology Definition Metamodel)
- ❑ RDF and RDFS – MOF supported



ESSEO **Model Driven Engineering for Context-aware Development**

Methodology

```

graph TD
    A[PIM metamodel specification] --> B[Context metamodel specification]
    B --> C[Match and Adaptation]
    C --> D[CPIM]
    D --> E[Mapping and transformation]
    E --> F[CPSM]
  
```

Context-aware Model Driven Development 13

ESSEO **Model Driven Engineering for Context-aware Development**

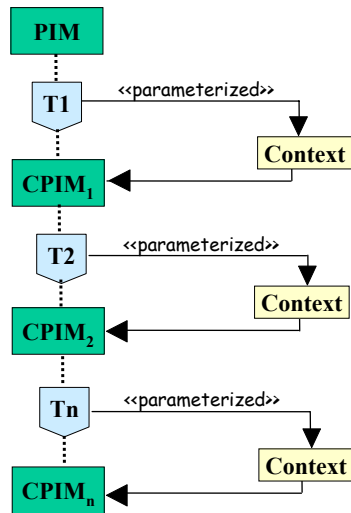
Methodology

```

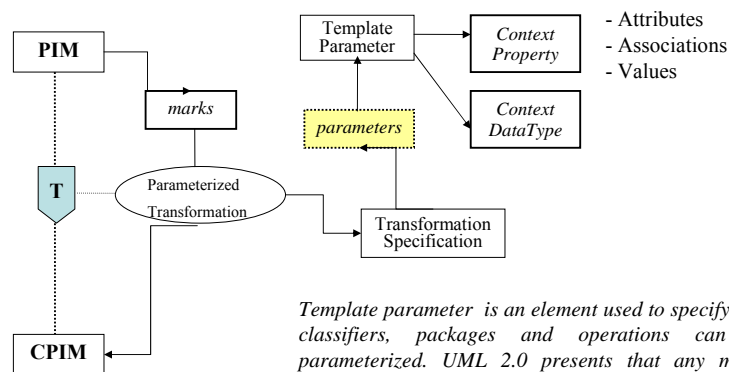
graph TD
    A[CPIM] --> B[CPIM Adaptation]
    B --> C{Final CPIM ?}
    C -- N --> B
    C -- Y --> D[CPIM_n]
    D --> E[Mapping and Transformation]
    E --> F[CPSM]
    F --> G[Transformation Engine]
    G --> H[Executable Model]
    H --> I(( ))
  
```

Context-aware Model Driven Development 14

Parameterized Transformation



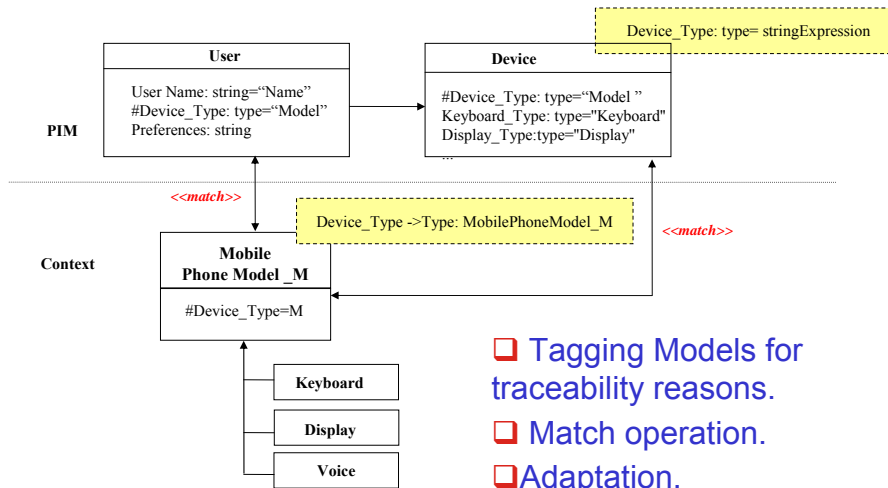
Parameterized Transformation



Template parameter is an element used to specify how classifiers, packages and operations can be parameterized. UML 2.0 presents that any model element can be templateable.

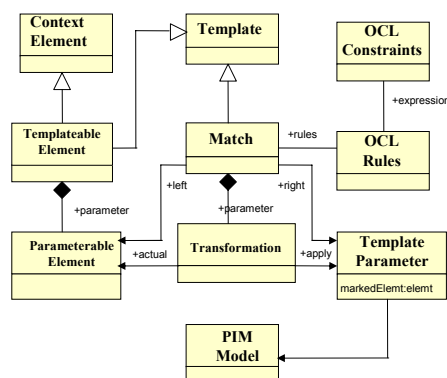
“A parameter specifies how arguments are passed into or out of an invocation of a behavioural feature like an operation. The type and multiplicity of a parameter restrict what values can be passed, how many, and whether the values are ordered” (UML 2.0)

Parameterized Transformation - Example



- ❑ Tagging Models for traceability reasons.
- ❑ Match operation.
- ❑ Adaptation.

Match Metamodel



```

match (TemplateParameter.markedElement.
allparameters ->
collect (CtxElement |CtxElement.
ParameterableElement.
type (x,y,z) and
collect (MarkedElement |markedElement.
TemplateParameter.
type (x,y,z) and
select -> (CtxElement |CtxElement.
ParameterableElement.
type (x,y,z) isCompatiblewith
MarkedElement |markedElement.
TemplateParameter.
type (x,y,z) ]
result= select match{
correspondences->collect (c|match.
correspondences.at(index)) }
Self -> transform (TemplateParameter |
TemplateParameter.Type (x,y,z).markedElement->
includes (parameterableElement)) ]
  
```

x, z = semantical elements, data types and resources
y = relations

Conclusion and Ongoing Works

- ❑ MDD for Context-aware Development.
- ❑ Advantages: different abstraction levels, individual and independent models, transformation techniques.
- ❑ Match and Adaptation.
- ❑ Target platform: Service Oriented Context-aware Architecture
- ❑ Eclipse *plugin* for context definition and adaptation.
- ❑ *Match* algorithm implementation.

Thanks
for your
attention !