MetaDone, a flexible metaCASE to support evolution

V. Englebert
P. Heymans
Plan

1. Motivations with 4 Scenarios
2. The MetaCASE Approach
3. MetaDone: a flexible metaCASE
4. Scenarios support by MetaDone
Part 1
Motivations: 4 Scenarios

Which problems metaCA SE technologies can address?
The Vision

Models \( \{M_1, M_2, \ldots, M_n\} \)

Business Needs

Software System

Models supports understood as
Model Evolution: several scenarios

- Evolution can occur at many levels: code, models, language
- Next slides show four scenarios that explicit how and why models evolve
Scenario #1

- The business changes, one model/software must be modified.
- Examples
  - Add one attribute to a class in a data model
  - Change the statechart of a class
  - Add a parameter to a method in the code
Scenario #2

- The business changes, one model must be modified, but this model must be synchronized with another model.
- Example
  - Add one attribute to a data model, but the behavior of the classes is constrained by statecharts that reference the classes' attributes.
Scenario #3

- The business changes, the metamodel must be enhanced to take into account new specificities
- Examples
  - EAI integration needs to add ontological annotations to class diagrams
  - The use of a new middleware entails to adapt the component model (PSM).
Scenario #4

- The business changes, new metamodels (languages) must be used to describe the new needs
- Example
  - B2B integration of a standalone IS entails the use of choreography, coordination, interaction, composition, … languages
  - Use a DSL for describing the ACL
Part 2
The MetaCASE Approach

But what is a metaCA SE?
The metaCASE approach

- **(Meta)*Models are evolving**
  - Previous scenarios show that software engineers will be confronted with not only evolving software, but also with evolving models and metamodels.

- **CASE tools have “constant” metamodels.**
  - Classical CASE tools are not able to support any modification of the metamodels. Hence, it is difficult to adapt them to manage the model evolution itself.

- **MetaCASE tools are a solution.**
  - MetaCASEs, whose first objective is to let the engineers to define their own metamodels, are better suited for that purpose.
A metaCASE allows software engineers to use dedicated CASE tools, formalized by abstract operational definitions that can be changed dynamically.
State Of the Art

• Several tools exist: MetaEdit+, GME, Metis, ConceptBase, Protégé, …
• They suffer from at least one restriction

  A. 1-1 relation between the metamodel definition and the concrete syntax definition.
     • the concrete notation influences the way the metamodel is defined!

  B. Metamodels and models definitions must be disjoint
     • A metaobject/object can not belong to several metamodels/models at the same time
     • Example: the name of an operation in a Sequence Diagram/Static Diagram/Statechart

  C. Strict separation of the modelling levels
     • An object cannot belong to a model and a metamodel at the same time
     • Harel is both the creator of the statechart metamodel and the idle state.

  D. Only the model level of the MOF pyramid can really be edited
Part 3
MetaDone: a Flexible Approach

“A nimals know, but man knows that he knows”

MetaDone proposes a new approach

I. define the metamodels with simpler and more expressive concepts than, i.e., the MOF.
   - concepts are metaobjects, metaproperties, metarelations, and metamodels, where everything is a metaobject.
   - refinement relationship between metamodels

II. break the strict separation of the modelling levels
   - one concept can occur at the metamodel and the model levels at the same time.
   - a model can be upgraded in order to become the metamodel of another model.
   - the metametamodel is the metamodel of itself, and can be interpreted either as a model or a metamodel.
   - etc
MetaDone

III. Reification: the tool is completely “model-driven”.
   • Every observable behavior or aspect of the tool should be the interpretation of some first-class model
     • GUI, Method, Constraints, Notations, Documentation, …
   • MetaDone is this fully customizable

IV. Engineers can define as many concrete notations for their metamodels (graphical or textual) as they want
   • the definitions of the concrete notations are themselves first-class objects
   • the concrete notation is itself a first-class object
Illustration

a metamodel MM

a model M1 of MM

model M2 as a refinement of M1
Part 4
Scenarios Support by MetaDone

“I can play any scenario!”
An unknown actress
MetaDone & Evolution

- **Metamodels definitions can evolve** naturally while keeping their models coherent
  - concepts can be added/retracted/modified
  - the refinement of metamodels enables variation points in the repository
- The method can evolve by **adding new metamodels** to existing metamodels
- **Concrete notations can evolve**
- **The metametamodel can itself evolve**
- As everything is a first-class citizen, the **evolution could become an observable artefact** that we could investigate/measure/…
MoVES

• Possible connections with MoVES:
  • WP2 Modelling and restructuring
    • « this work package will focus on modelling languages, meaning those languages whose primary role is to model software systems, at different stages and under different viewpoints, rather than produce a final implementation. Our goal is to push the state-of-the-art in devising or extending modelling languages, [...] »
  • WP4 Consistency and coevolution
    • « The goal of this work package is to study and address the problem of inconsistencies arising between models expressed in different modelling languages. »
Future Work

• Continue to port the $\alpha$-prototype$_{C++}$ to the operational framework$_{Java}$
• Develop and improve the GUI
  • challenge: how to reduce the cognitive effort of the engineers ?
• Improve the support of (meta)*model evolution ?
  • collaborative work
  • traceability
  • versioning
  • “post-mortem” integration (cfr. our experiences in databases & ontologies)
• Transformation Language
• No metaCASE addresses methods (vs models)
  • if metamodels evolve, the methods should also evolve !
• Validation
  • Case studies, usability evaluation, …
References

Thank you!
Any questions?

Contact:
- ven@info.fundp.ac.be
- phe@info.fundp.ac.be
- Nicolas Genon
- PRC Henri Tudor Lux

University of Namur
http://www.fundp.ac.be

Computer Science Faculty
http://www.info.fundp.ac.be

PRECISE group
http://www.software-engineering.be/