PROCESS BEHAVIOR META-MODELING FOR THE DERIVATION OF EXECUTION ENGINES

Sana Damak MALLOULI
Centre de recherche en informatique, Paris I University Pantheon-Sorbonne

General context

Engineering of Software tools is a complex, error prone and time consuming task. In previous research work [Rolland et al. 88], [Souveyet & al. 98], [Si-Said & al. 96], software tools for event-oriented and goal-oriented modeling formalisms were engineered in an ad-hoc manner.

- Building software tools according to model-driven engineering (MDE) overcomes some of these drawbacks.
- Meta-models become essential artifacts as they are expected to describe the structure, the semantics and the future usage of the models to be manipulated by the tool.

The resulting software tools:
- Suffer from maintainability and portability problems
- Remain temporary demonstration prototypes (non reusable)
- Could not evolve to support new modeling languages

- Meta-modeling practice is usually restricted to specifying the static structure of models
- There is a lack of the behavioral specification in existing meta-model languages (Kermit, MOF, Ecore, GOPR,...)
- Problem: the operational semantics expression of models
The difficulty of properly expressing and exploiting dynamic behavior of process models at meta-model level handicaps the automatic derivation of execution engines.

Behavior meta-modeling

Proposal: Expressing a process model operational semantics by specifying an enactment engine architecture and using a behavior meta-model

Method Engineer point of view

- Structural view of the process meta-model
- Dynamic view of the tool architecture
- Global vision of the tool architecture

Design point of view

- OO Meta-model (UML)
- Consistency rules
- Event meta-model (Remora)
- Conceptual transformation rules
- UML specification of the tool

Implementation point of view

- DTD of static specification
- DTD of dynamic specification
- Transformation rules in ATL
- DTD of tool architecture

References

Application to the Map formalism

The Map engine architecture

Tool environment representation

Tool specification notations

Execution Engine

Execution in UML code generator

Map operational semantics

The Map behavior representation

Transformation of the behavior meta-model into an executable object-oriented application based on the publish/subscribe pattern:

Rules For external events

Rules For internal events

- In order to derive an execution tool from these meta-specifications, we define transformation rules that target Java platforms
- Because of the dynamic and interactive nature of the behavior model, we rely on the publish/subscribe development patterns.

- Our contribution can be exploited in several contexts where is needed to specify and support new process modeling languages.
- To overcome gaps in the expression of a process modeling language execution semantics, it is possible to improve such expression and enhance execution engine quality through behavior meta-modeling by representing the process semantics in a declarative format and with graphic notation.
- The execution tool is derived with less programming effort and less error risk.

Madness 2014, Paris, France