Role-based Multi-Purpose Workflow Engine.

Sebastian Richly
Sebastian Götz
Uwe Aßmann
Dirk Habich

Funchal, Madeira, Portugal, 08.06.2010
Motivation

- big and fragmented WfMS ecosystem
  - many specializations exist

- new requirements, standards or domains
  \(\Rightarrow\) adapt engine

- integration currently on a per-case base with high effort

- lack of runtime flexibility, that is
  - ability to adapt unanticipated
The features of our approach:

1. Support for all petri-net based process languages
2. Dynamic extension of new element types
3. Runtime flexibility
4. Integration of different process languages
5. Mix of domain concepts
1) Support for all petri-net based process languages

- Engine based on **Workflow Nets** [1]
  - restricted Petri Nets
    - requires input and output place
    - must not have dangling places
  - able to represent all petri net based workflow languages
  - prerequisite for integration of different workflow languages (*least common multiple*)
  - provides formal techniques of petri-nets
Utilization of **Role-based Programming** [2]

- Roles as extension to the object-oriented paradigm
- Concept stems from theater
  - objects start and stop to play roles over time
  - objects are able to play multiple roles concurrently
  - played roles change behaviour and structure of player

```
john : Person
  age := 21
  name := „John“

: Student
  stud.id := 3013737

: Employee
  salary := 400€

: University
  name := „TUD“

: Company
  name := „Ex. Inc.“
```
– 5 different workflow aspects exist[3]:

- Functional (What)
- Operational (Applications)
- Informational (Data/Flow)
- Organizational (Who)
- Behavioral (Control Flow)
Workflow Aspects in **Role Space**:

- **Behavioural Context**
  - Abstract Net Element
    - Transition
    - Edge
    - Place
    - Token
  - Workflow Net

- **Manager** and **Executor**
- **Informational Context**
  - Data-Container
    - Data
      - Constant
      - Value
      - Reference
      - Var
    - DataFlow

- **Organisational Context**
  - Organisation
    - Recourse
    - Sub-Organisation
  - Human
  - Computer

- **Operational Context**
  - Operation
    - Service
    - Bundle
    - WebService
    - JavaClass

- **Functional Context**
  - Function
  - Condition
Credit Card Example:

Role-based Multi-Purpose Workflow Engine
2) Dynamic extension of new element types

- New Role-Bundle
  - OSGi Bundle

- Behavioural Context
  - Abstract Net Element
  - Workflow Net
  - Transition
  - Edge
  - Place
  - Token

- Manager
  - Control

- Executor
  - Control

- Informational Context
  - Data-Container
  - Data
  - DataFlow
  - 2
  - Constant
  - Value
  - Reference
  - Var

- Organisational Context
  - Organisation
  - Recourse
  - Sub-Organisation
  - Human
  - Computer

- Operational Context
  - Operation
  - Service
  - Bundle
  - WebService
  - JavaClass

- Functional Context
  - Function
  - Condition

TCoB’10, 08.06.2010
Role-based Multi-Purpose Workflow Engine
Slide 9 / 15
3) Runtime flexibility

- proactively decide instead of react after failure
- change role set of running processes
4+5) Integration and Intermixture

- possible due to workflow nets
  - language importer written
  - First control flow concepts are mapped
  - Second resources and activities are mapped
- mix of domain concepts possible
  - Tasks can play multiple roles at the same time
  - even if these roles stem from different imported languages!
03 Implementation

http://141.76.65.194/OSPP
- **XPDL** for interoperability between workflow languages, defined by WfMC

- **XRL**[4] (eXtendible Routing Language) also based on Workflow Nets, changes only at design time (not runtime)

- **AO4BPEL**[5] allows to weave in tasks at runtime, but is tailored to specific languages

- **XSLT transformations** for interoperability of workflow languages, like in [6]
1. Support for all petri-net based process languages
   ➔ because based on workflow nets

2. dynamic extension of new task types
   ➔ tasks start to play additional roles

3. runtime flexibility
   ➔ proactively decide instead of react on failure
   ➔ extend/change at runtime using role bundles

4. integration of workflow languages
   ➔ due to workflow nets
   ➔ domain concept mix possible due to roles
Thank you!

Any Questions?


