ReFFlow: Reusable Flexible WS-flows

Databases and Distributed Systems Group
Dimka Karastoyanova, Alejandro Buchmann
dimka@gkec.tu-darmstadt.de, buchmann@informatik.tu-darmstadt.de

ReFFlow Project

- **Motivation**
  - Missing specifications
    - Standard methodology for development and execution of Web Service compositions
    - Unified WS compositions model
  - Inadequate support for WS-flow adaptability
  - Lack of execution engine and supporting tools for WS-flows with built-in adaptability

- **Procedure for Development and Execution of Process-based Composite Web Services**
  - Based on WS-flow life-cycle
  - Phases prescribe approaches to address different aspects of a process definition
  - Promotes the creation of unified WS-flows meta-model with built-in adaptability
  - Automation of WS-flows development using templates

- **Meta-Model:**
  - Extends existing models
  - Constructs:
    - Dynamic selection and invocation of WS instances
    - Dynamic changes of process schema
    - WS types
    - Process logic
    - Selection policies
    - QoS parameters
    - Independent of implementation approach
    - Promote WS-flow standardization and portability

- **Build time**
  - Development automation
  - Based on common WS-flow model
  - Reuse of process definitions

- **Run time**
  - Desired features:
    - Process adaptability and flexibility
    - Users control the adaptation of processes
    - Details of the implementation approach remains hidden (transparent)

Platform

- 

Methodology

- **Process template modeling and assembly**
  - Model and assemble templates and parameterized processes
    - WS-flow templates: design patterns, domain-specific templates, coordination protocols roles
  - Use meta-model constructs
  - Produce abstract process definitions
    - Avoid any references to specific WS instances and to WSs portTypes
  - Add additional business logic

- **Process definition generation phase**
  - Transform the templates and parameterized processes into executable process definitions
  - Use traditional meta-programming techniques
    - Code generators, transformations of XML documents

- **Compile and pre-processing time**
  - Optional - depend on the target definition language

- **Deployment**
  - Enrich WS-flow definitions with
    - Execution environment specific data
    - Details about the participating WSs
    - Binding information of WS-flow

- **Execution time**
  - Process instances are created and executed
  - Adaptability (flexibility) supported by the system catalogue of the process engine and the extension activities

- **Post-run time**
  - Analyze the process progress and logic
  - Use information gathered during run time
  - Change process schema accordingly

Future Work

- **Tools**
  - WS-flows templates and model repository
  - Process modelling tool
  - Support coordination protocols
  - Transform definitions into multiple languages
  - Instance browser

- **Engine Implementation**
  - Model extension constructs for built-in flexibility

Methodology