Web Service Migration with Migration Decisions Based on Ontology Reasoning

M. Mohanned Kazzaz and **Marek Rychlý**

Department of Information Systems
Faculty of Information Technology
Brno University of Technology
(Czech Republic)

12th International Scientific Conference on Informatics,
5 – 7 November, 2013

Outline

Service Context in Service-oriented Architecture

- during its life-time, a service is put into a specific context
  - by its integration into an architecture (design-time),
  - by its deployment into an IT infrastructure (deployment-time),
  - by its actual runtime environment (run-time).
- design-time and deployment-time contexts are essentially static
  (it is well-understood how they affect service functionality and quality)
- run-time context is highly dynamic
  (it may affect functionality and quality of a service in generally unpredictable way)
- how to assure required functionality and quality of a service despite evolution of its runtime context?

Service Context-awareness and Service Migration

- to cope with the dynamic run-time context of a service we can
  - control the service's runtime environment,
    (may not be possible, especially in highly volatile environments)
  - analyze all possible states of the environment
    and verify that the service will always work as expected,
    (requires limited runtime environments, cannot be unpredictable)
  - make the service aware of current runtime context.
    (make the service adaptable to some of potential environmental changes)
- a context-aware service may react to changes in the context
- service migration is a reaction to an inconvenient runtime context
  1. the inconvenient runtime context of a service is detected,
     (by the context-aware service or by its observer; e.g., in the case when a service's current provider is not suitable for further providing of the service)
  2. the service is (re)deployed to different and more suitable provider
     (the process controlled by the service itself or by a migration controller)

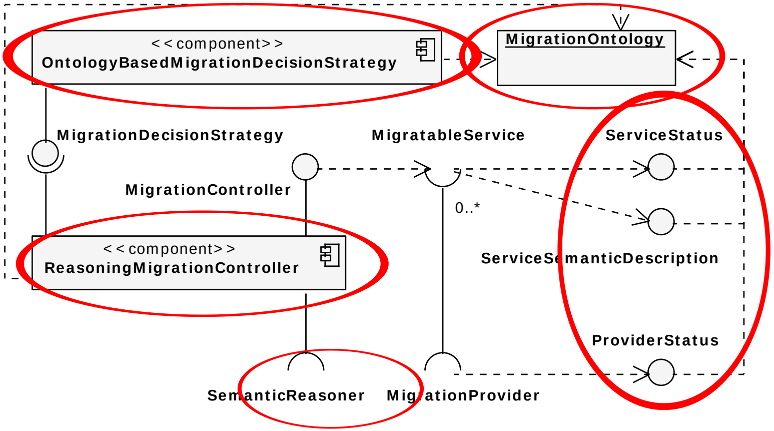
The Web-service Migration Framework

- the framework defines Web-service interfaces for
  - observing the runtime context of services,
    (e.g., an utilization of the services, a state of they providers, a state of network connections between providers of cooperating services, etc.)
  - setting runtime context requirements by services and providers,
    (services require particular runtime properties of their potential providers, providers require particular runtime props. of potentially provided services)
  - controlling the service migration process.
    (to deploy of a service into a new provider, to transfer the service's state between the deployments, to undeploy the service from its original provider)
- the framework implements algorithms for
  1. detection of inconvenient runtime contexts,
     (of currently provided services)
  2. selection of a new potential provider,
     (where a vulnerable service will be migrated)
  3. migration of the service into the new provider.

Ontology Reasoning in Migration Decisions
Motivation

- the goal has been to enhance the framework
  by introducing the ontology reasoning into migration decisions
  - detection of inconvenient runtime contexts,
  - selection of a new potential provider.
    (the new provider has to ensure suitable runtime context)
- to propose
  - an ontology for the runtime context in general,
  - provable customizable statements describing the suitable context.
- these allow to describe the runtime context and migration decisions in more general ways and formally

Architecture of the WS-Migration Framework



- the ontology for service runtime context, context state reporting
- migration decisions by ontology reasoning, migration control
- external ontology reasoner

Service Runtime Context Ontology

- basic concepts (device, migratable service/provider, migration candidates)
- designed in SADL (Semantic Application Design Language)
- implemented in OWL 2/RDF
- based on OWL-S (Service, Profile)

```
Device is a top-level class.
/* a provided service is an OWL-S service provided by a service provider */
ProvidedService (alias "provided service") is a type of
OWLS_Service:Service.
/* a migratable service and its specification by an OWL-S profile */
MigratableService is a type of ProvidedService.
MigratableServiceProfile is a type of OWLS_Profile:Profile.
relationship of MigratableService to MigratableServiceProfile is
OWLS_Service:presentedBy.
CandidateForMigrationService is a type of MigratableService.
/* a service provider */
ServiceProvider (alias "service provider") is a top-level class
described by hostname with a single value of type string
described by protocol with a single value of type string.
protocol of ServiceProvider always has value "SOAP".
relationship of ServiceProvider to Device is hostedBy.
/* types of ServiceProvider participating in migration decision process */
CandidateOriginServiceProvider is a type of ServiceProvider.
CandidateDestinationServiceProvider is a type of ServiceProvider.
/* attributes indicating empty preference rules set */
noPreferenceRules describes { MigratableService or
CandidateDestinationServiceProvider } with a single value of type
boolean.
/* a migration decision, a mapping of services which will/can be migrated to
other providers */
MigrationDecision is a top-level class
described by migratedService with a single value of type
CandidateForMigrationService
described by destinationProvider with a single value of type
CandidateDestinationServiceProvider.
```

Service/Provider's Preferences in Migration Decisions

- services/providers require particular runtime properties of their potential providers/services, respectively
- services/provider publish the requirements as their "preferences"
- the preferences are described as logical expressions (fragments of Jena rules for future migration decision reasoning)
- from these preferences, Jena rules are generated to reason
  - all services satisfied with runtime context of a given provider, (reasoned as predicate "possibleProvidedService")
  - all providers able to provide a given service with its runtime context. (reasoned as predicate "possibleDestinationProvider")
- these "preference rules" generated Jena rules are utilized further in migration decisions as follows. . .

Migration Decisions Rules

1st to identify a service to migration (due to inconvenient runtime context according to service's or provider's requirements)

2nd to select a provider where the service will be migrated (where its runtime context is convenient)

- designed in SADL
- implemented as Jena Rules

```
/* 1st step of migration decision making */
Rule LookForCandidateForMigrationServiceDueToServicesPreferences
      given  service is a MigratableService
             origin is a CandidateOriginServiceProvider
      if     origin provides service
             origin is a CandidateDestinationServiceProvider
             service has possibleDestinationProvider not origin
      then   service is CandidateForMigrationService.
Rule LookForCandidateForMigrationServiceDueToProvidersPreferences
      given  service is a MigratableService
             origin is a CandidateOriginServiceProvider
      if     origin provides service
             origin is a CandidateDestinationServiceProvider
             origin has possibleProvidedService not service
      then   service is CandidateForMigrationService.
/* 2nd step of migration decision making */
Rule LookForMigrationDestinationsForEachMigratingService
      given  service is a CandidateForMigrationService
             origin is a CandidateOriginServiceProvider
             destination is a CandidateDestinationServiceProvider
      if     origin provides service
             origin is not destination
             service has possibleDestinationProvider destination
             destination has possibleProvidedService service
      then   a MigrationDecision
                   with migratedService service
                   with destinationProvider destination.
```

Summary and Future Work

- Service runtime context has been described by OWL 2/RDF ontology.
- Services and providers publish migration preferences as logical expressions.
- From the preferences, Jena rules are generated describing inconvenient runtime contexts.
- Jena is utilized for reasoning based on the ontology/facts of current runtime contexts and preference rules to make migration decisions.

Future work

- Prototype implementation of the framework with ontology reasoning.
- Evaluation and optimization of the ontology reasoning in migration decisions.

Thank you for your attention!

Marek Rychlý

<rychly@fit.vutbr.cz>