DynamCoS: a Framework for the Dynamic Composition of Services
TTT 15th of January 2009

Eduardo Silva
Luís Ferreira Pires
Marten van Sinderen
(Jorge Martínez López)

Outline

- Dynamic service composition (life-cycle)
- DynamCoS framework (and prototype)
- Evaluation and validation
- Conclusions and future work

IT Organisations and Users

“Gartner Highlights Key Predictions for IT Organisations and Users in 2008 and Beyond” says:

- “By 2012, at least one-third of business application software spending will be as service subscriptions instead of as product license. With software as service (SaaS), the user organisation pays for software services in proportion to use... Endorsed and promoted by (Oracle, SAP, Microsoft, Google, Amazon)…”

- “By 2012, 50 per cent of travelling workers will use new classes of Internet-centric pocketable devices at the sub-$400 level; and 30 per cent will have applications that can be accessed from everywhere. A personal workspace that encapsulates a user’s preferred work environment…”

New ideas... New challenges:
- Software as service (SaaS)
- Internet-centric pocketable devices
- server and Web-based applications

These trends trigger new ideas and research challenges, namely service composition

(Dynamic) Service composition...

- Where and how ?...
- Manual or can be automated ?...
- Design or Runtime ?...
- ...

Dynamic Service Composition
Dynamic Service Composition Life-cycle

Automatic composition of services (Runtime and Design time) 
- Service creation
- Service deployment
- Service registry
- Service discovery
- CLM Construction
- Graph-based service composition
- Resulting Graph Compositions

Service Creation and Publication

Semantic Annotations

- Services and service requests semantic Annotations:
  - Inputs: information space before service execution
  - Preconditions: state of the world before the execution of the service ("has a bank account")
  - Outputs: information space after service execution
  - Effects: state of the world after the execution of the service ("ticket has been bought")
  - Goals: representations of an objective for which fulfillment is sought through the execution of the service
  - Non-functional properties

Service Creation and Publication

AlertService
- Goal: Send Alert
- #{enzota}49:#{enzota}61:#{enzota}70
- Hospital: #{enzota}49:#{enzota}50:#{enzota}53
- Service Publication
- Service composition
- Service deployment
- Service registry
- Service discovery
- CLM Construction
- Graph-based service composition
- Resulting Graph Compositions

Service request
Example scenario

- Let's imagine that a service developer is asked to create a new service for a health insurance company. This service will be provided to people with health problems, allowing to assist them in emergency situations. The service allows the patient to trigger an alarm, which starts the service locating the patient based on his/her telephone location; based on the location information the service shall find the nearest hospital, and then call for an ambulance to carry the patient from his/her location to the hospital. Additionally the service has to notify a contact person of the patient, giving him/her the necessary information to locate the patient.

Service request

- [Diagram showing a service request process]

Service discovery

- [Diagram showing a service discovery process]

(Goal-based) Service discovery

- [Diagram showing goal-based service discovery]
A service composition is a sequence of services that have semantically related interfaces (outputs and inputs of consecutive services are related)

\[ S_i \text{ input (In}_iS_i) \]
\[ S_i \text{ output (Out}_iS_i) \]

(Semantic) Causal Link

\[ S_j \text{ input (In}_jS_j) \]
\[ S_j \text{ output (Out}_jS_j) \]
Conclusions and Future work

- Conclusions:
  - Current trends on computer and communication systems trigger new ideas for service creation and delivery, namely Dynamic Service Composition (DSC).
  - We propose a semantics-based framework, DynamiCoS, to address all the required phases of the DSC life-cycle.
  - DynamiCoS is modular and language-independent.
  - We have implemented a prototype and evaluated it, concluding that it provides a "real-time" response to automatic service composition.
  - Semantic reasoning is still a very heavy task.

- Future Work:
  - Extend service discovery to service composition time.
  - Support other service description languages.
  - Use non-functional properties and user context in the DSC process.

Thank you!
Any questions?

Email: e.m.g.silva@cs.utwente.nl
Web: www.cs.utwente.nl/~goncalvesem