Behavioral Contracts and Service Substitutability: A Contribution to Dependable SOA

Haldor Samset and Rolv Bræk,
Norwegian University of Science and Technology – NTNU,
Department of Telematics

NTNU, June 2008
What is a service?

A service is: *an identified functionality aiming to establish some goals/effects among collaborating entities.*

Captures:
- end user services
- active services
- passive services
- component interfaces (Web Services, CORBA, JINI, …)
- layered functionality (ISO OSI)
Service fundamentals:

- Service is *functionality*; it is behavior performed by entities.
- Service imply *collaboration*; it makes no sense to talk about service unless at least two entities collaborate.
- Service behavior is *cross-cutting*; it imply coordination of two or more entity behaviors.
- Service behavior is *partial*; it is to be composed with other services.
Service modeling using UML 2 collaborations

- Matches the concept of service: Collaborative; Cross-cutting; Partial; Functionality
- Can model services separately in terms of role structures and behaviours
- Allows flexibility in binding roles to classes
- Require conformance between roles and classes
- Can model interfaces and contracts as two-party collaborations
Collaboration as behavior contract: example

```
<<basicService>>
StockQuote

<<serviceRoleStateMachine>>
requestor
: StockQuoteRequestor

<<signal>> StockQuoteNotification
<<signal>> AckStopSubscription

<<serviceRoleStateMachine>>
provider
: StockQuoteProvider

<<signal>> SubscribeStockQuote
<<signal>> StopQuoteSubscription

sm requestor

! SubscribeStockQuote

receiving

? StockQuoteNotification

! StopQuoteSubscription

stopping

? StockQuoteNotification

? AckStopSubscription

sm provider

? SubscribeStockQuote

notifying

! StockQuoteNotification

? StopQuoteSubscription

stopping

! AckStopSubscription
```
Collaboration as behavior contract:

- Two connected roles with
  - Static interfaces
  - Interface behaviors
- Connector properties
  - Asynchronous or synchronous
  - Bidirectional or unidirectional
- Modelchecked to ensure compatibility between roles
- Publishable using WSDL
Using contract roles to type interfaces

- Compatibility of contract roles modelchecked at design time
- Conformance with contract checked for each interface at design time
- Simple compatibility assurance at runtime
Conformance with contract

1. Project component behavior to interface behavior
2. Compare interface behavior with contract role behavior: are they equivalent or substitutable?
An extended contract

```
<basicService>
StockQuoteAndNews

<serviceRoleStateMachine>
requestor
:StockQuoteAndNewsRequestor

<signal>StockQuoteNotification
<signal>StockNewsNotification
<signal>AckStopSubscription

<serviceRoleStateMachine>
provider
:StockQuoteAndNewsProvider

<signal>SubscribeStockQuote
<signal>SubscribeStockQuoteNews
<signal>StopQuoteSubscription
<signal>StopNewsSubscription
<signal>StopStockQuoteNews
<signal>AckStopSubscription
```
With additional behavior

Sq:StockQuote

Sq:StockQuoteAndNews
Safe substitution: equivalent reachable behavior

Sq:StockQuote

<<Substitutes>>

Sq:StockQuoteandNews

sm provider

? SubscribeStockQuote

notifying

! StockQuote

? StopQuoteSubscription

stopping

! AckStopSubscription

sm provider

? SubscribeStockQuote

! StockQuoteNotification

? SubscribeStockQuoteNews

quotes

! StockQuoteNotification

? SubscribeStockNews

! AckStopSubscription

? StopNewsSubscription

quotes &news

! StockNewsNotification

? StopQuoteNewsSubscription

stopping

stopping

? StopQuoteSubscription

! AckStopSubscription

! AckStopSubscription
Safe substitution

- Verified once at design time
- Simple checks at run time
Summing up

- Contracts are modelchecked collaborations
- Conformance ensured by projection and role comparison
- Run-time efficient compatibility assurance
- For active and passive services
- A basis for meaningful lookup