



DWFTT
September 13, 2007
Enschede, The Netherlands

Testing Web Services With Symbolic Transition Systems

Lars Frantzen
Radboud University Nijmegen
The Netherlands
lf@cs.ru.nl

Who's That?

- Ph.D. Student at the Radboud University Nijmegen
- Group of Frits Vaandrager
- Supervised by Jan Tretmans
- Just returned from an 18 month research visit at the CNR Pisa, Italy, group of Antonia Bertolino



The Others

- People involved in the presented work are (for instance):
 - Jan Tretmans
 - Tim Willemse
 - René de Vries
 - Antonia Bertolino
 - Andrea Polini
 - Guglielmo de Angelis

Agenda

- Symbolic Models
- Web Services
- Modeling Web Services
- Testing Web Services



Agenda

- Symbolic Models
- Web Services
- Modeling Web Services
- Testing Web Services



Symbolic Models

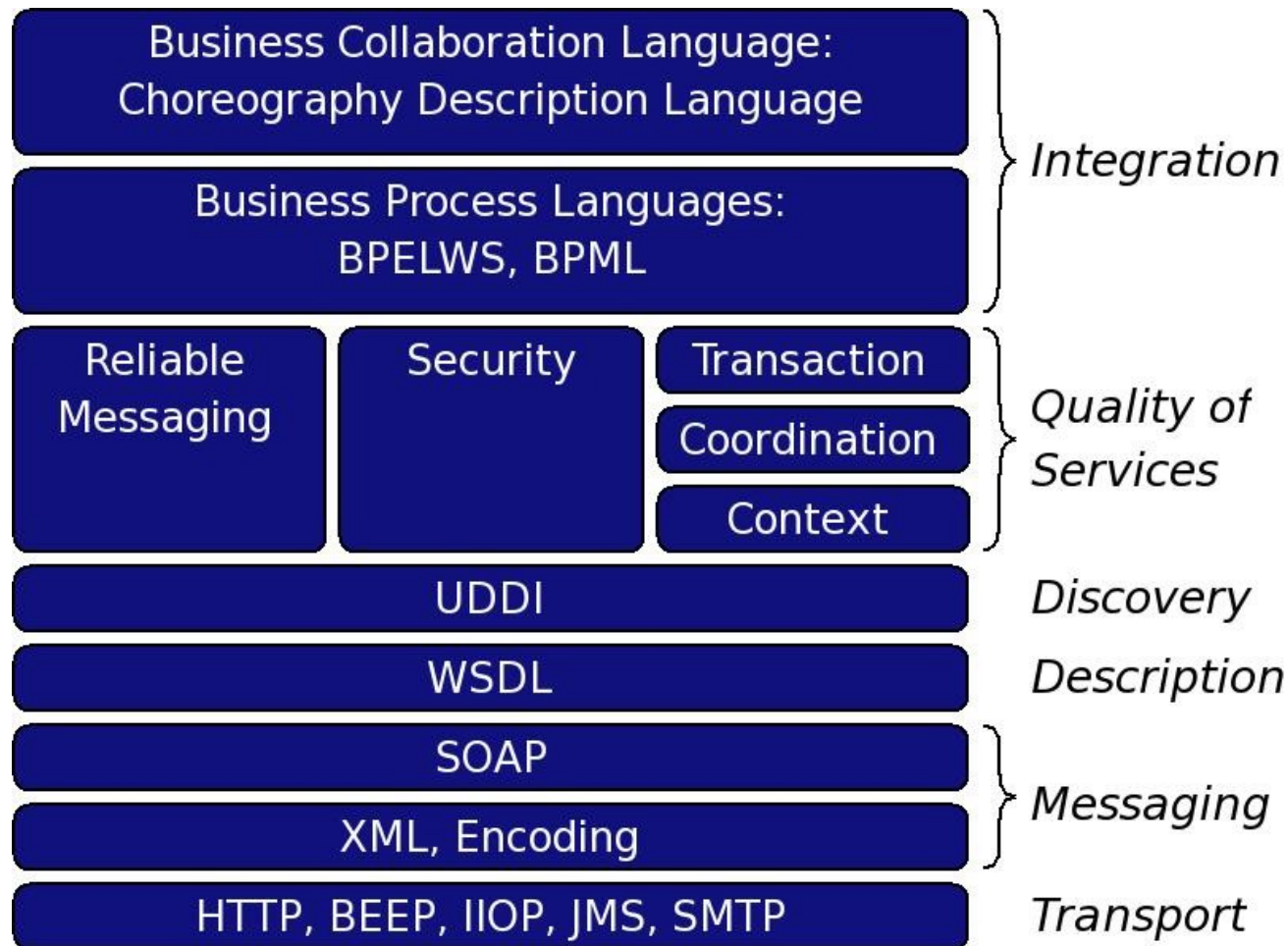
- Symbolic models symbolically deal with data
- Typical concepts: types, variables, guards, operations
- Motivation of symbolic testing: avoid **state space explosion** due to constructing semantical models (like LTS, FSM)
- A symbolic variant of ioco is ***sioco***
- In *sioco*:
 - Specification models are **Input-Output Symbolic Transition Systems (IOSTS)**
 - Implementation models are **input enabled IOSTS**

Agenda

- Symbolic Models
- Web Services
- Modeling Web Services
- Testing Web Services

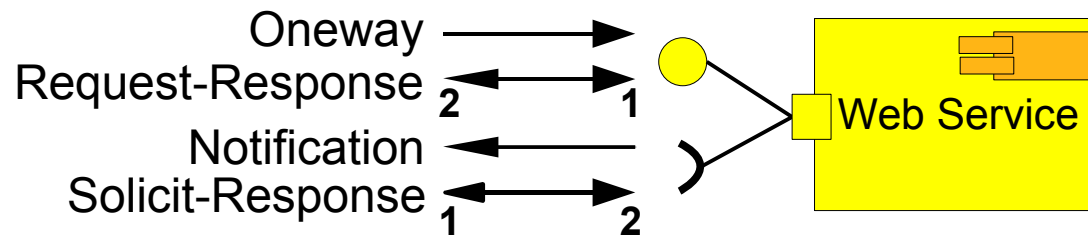
Web Services

- Relevant standards are:



Web Service Interfaces

- Web Service Interfaces are specified in the **Web Service Description Language (WSDL)**
- A set of **operations** is called a **port type**
- A port type is associated to a physical service **port**
- Four kinds of operations can be specified:



Orchestration & Choreography

- One goal of Service Oriented Architectures (SOA) is to allow for **composing services**
- A composition of services yields again an (**orchestrated**) service
- Relevant standard: **BPEL**
- Independent services may need to follow a **coordination protocol** to ensure correct communication
- Such a coordinated set of services is called a **choreography**
- Relevant standard: **WS-CDL**

Service Discovery

- A **directory service** offers mechanisms to publish and retrieve information about available services
- Services can dynamically bind to other services via directory services
- Services register and unregister dynamically
- Relevant standard: **UDDI**

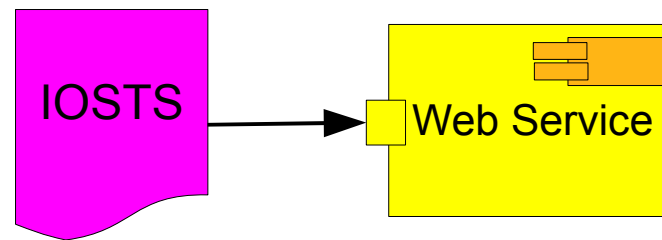


Agenda

- Symbolic Models
- Web Services
- Modeling Web Services
- Testing Web Services

Modeling Web Services

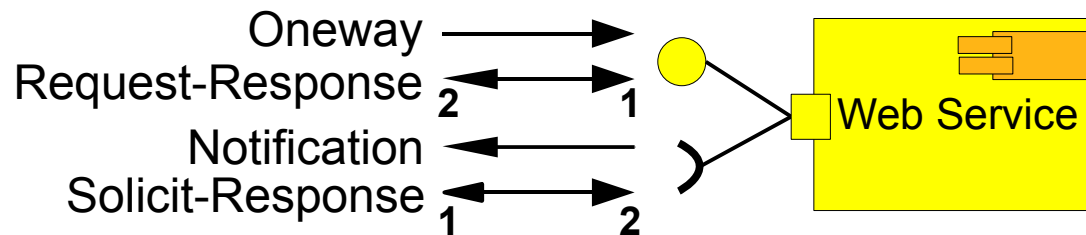
- A first natural choice: model a **port-type** via an IOSTS



- Possible **extension**: extend to **multi-ports**. This would allow for specifying *Coordination Protocols* and *Compositions*

Modeling Web Services

- IOSTS have input and output actions (aka *gates*)
- In **WSDL**:
 - Oneway → input message
 - Request-Response → input message followed by an output message
 - Notification → output message
 - Solicit-Response → output action followed by an input message



- For **IOSTS** we map:
 - Oneway → input action
 - Request-Response → input action followed by an output action
 - Notification → output action
 - Solicit-Response → output action followed by an input action

Modeling Web Services

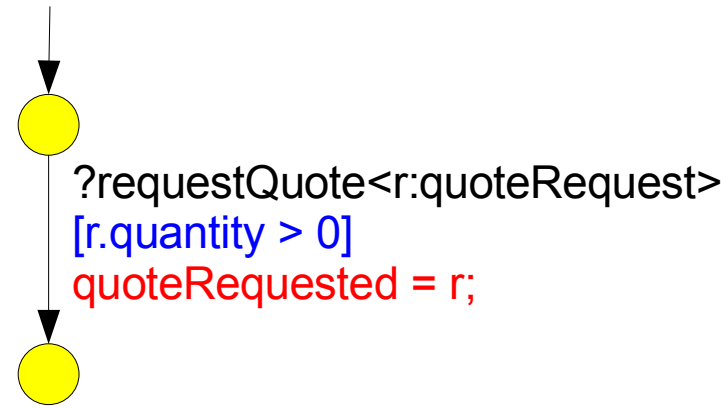
- A **Supplier** Web Service offering the operations:
 - requestQuote(r : **quoteRequest**) : **quote** (*Request-Response*)
 - orderGoods(ref : **integer**, quant : **integer**) : **boolean** (*Request-Response*)
 - makePayment(ref : **integer**, amount : **float**) (*Oneway*)
- Complex Types (Classes):

– **quoteRequest**
product : **product**
quantity : **integer**

– **quote**
price : **float**
product : **product**
quantity : **integer**
refNumber : **integer**
sum : **float**

product (Enumeration)
{foo, bar}

Modeling Web Services

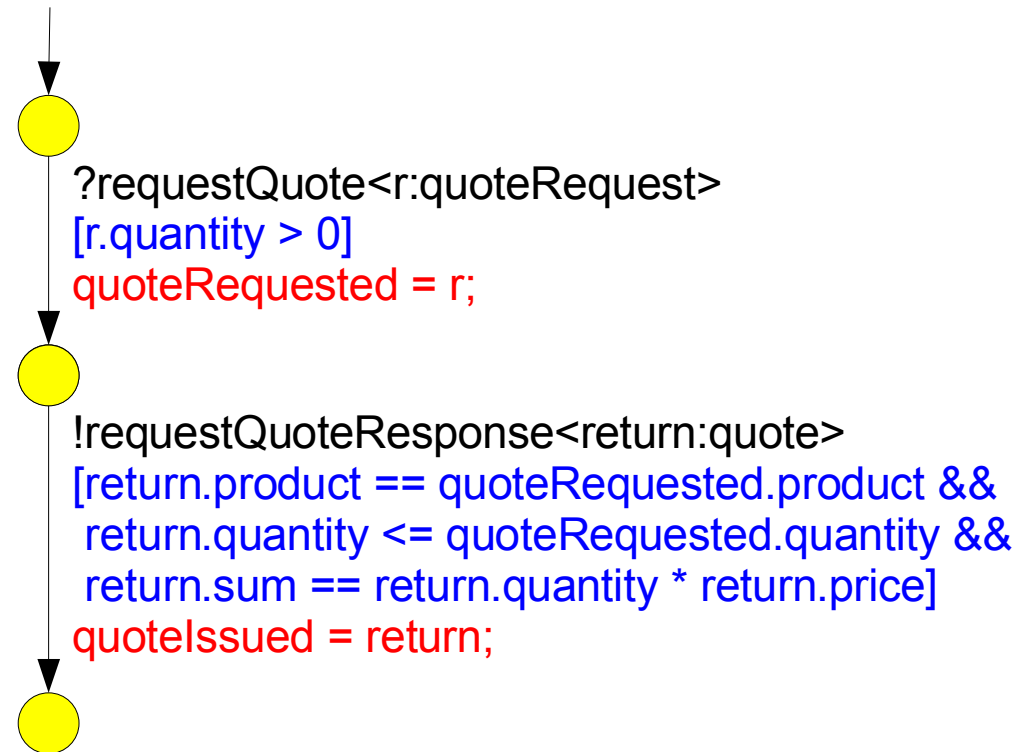


quoteRequest
product : product
quantity : integer

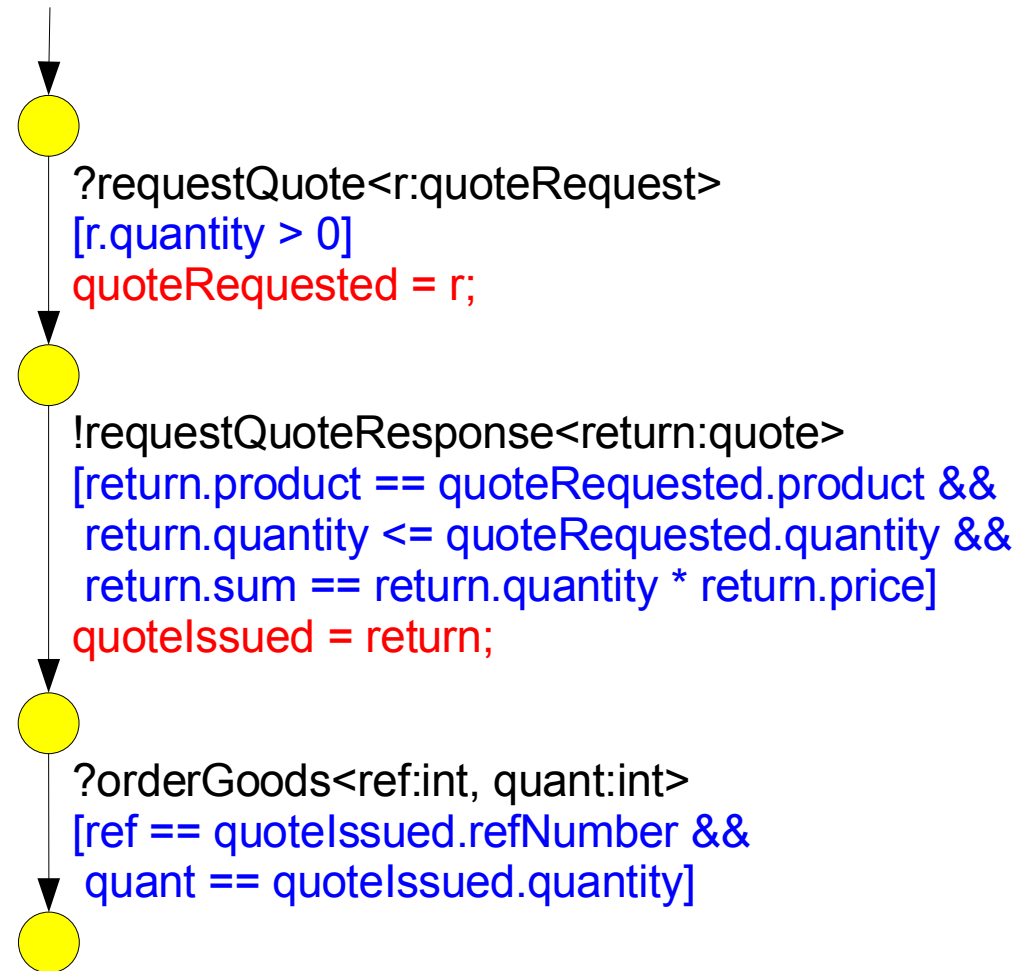
Modeling Web Services

quoteRequest
product : product
quantity : integer

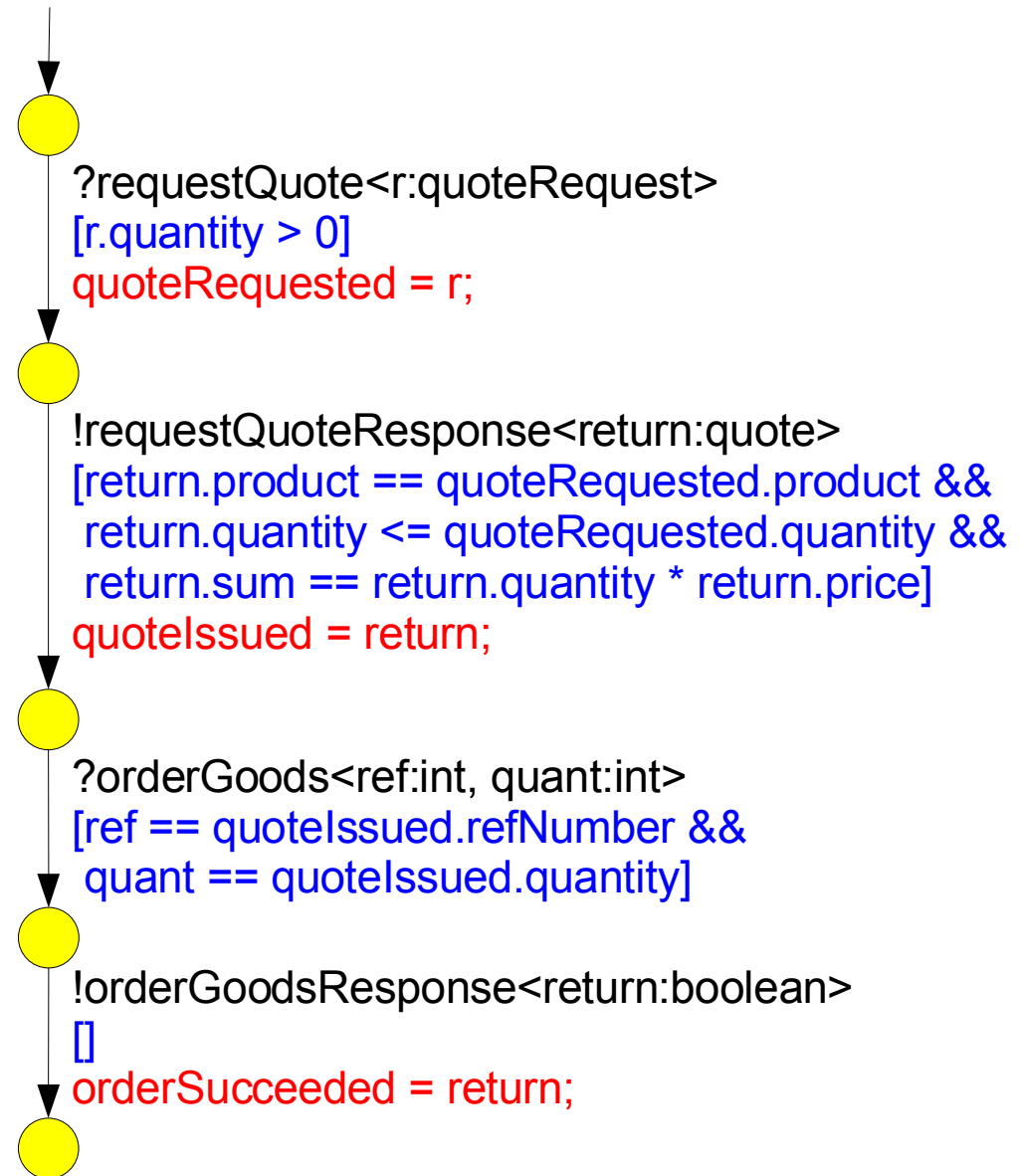
quote
price : float
product : product
quantity : integer
refNumber : integer
sum : float



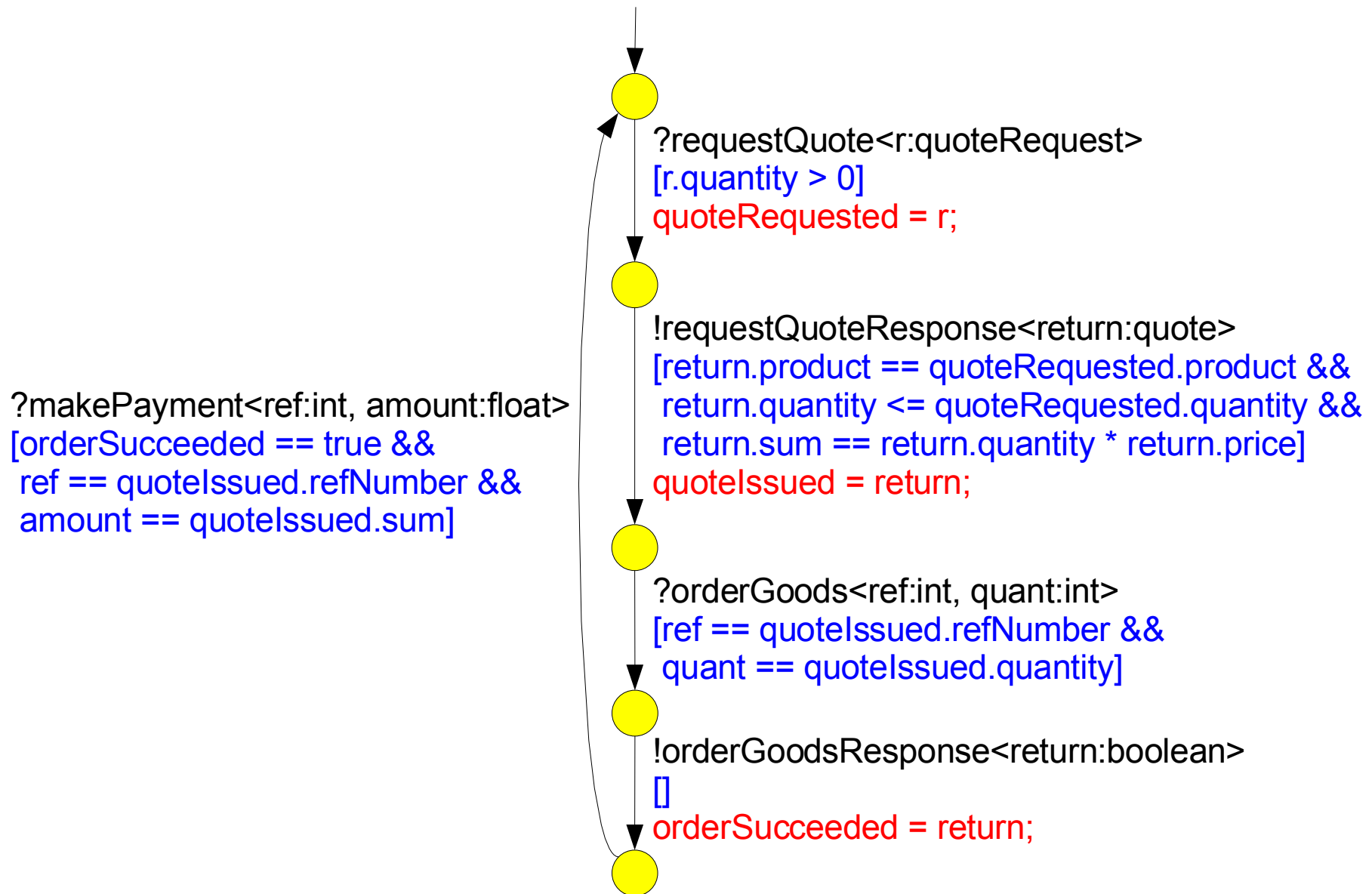
Modeling Web Services



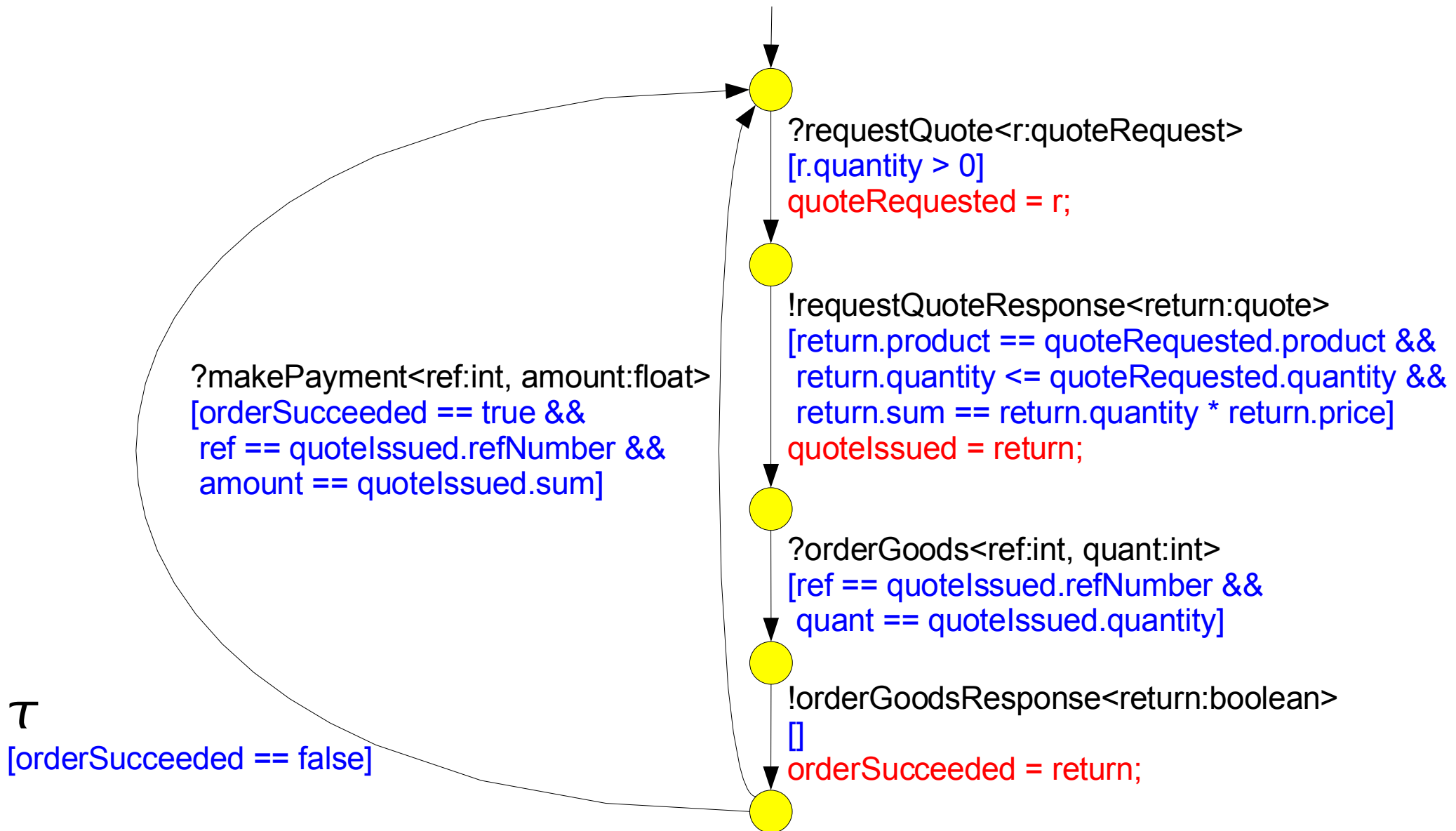
Modeling Web Services



Modeling Web Services



Modeling Web Services



Agenda

- Symbolic Models
- Web Services
- Modeling Web Services
- Testing Web Services

Symbolic Testing

- Doing symbolic testing does not come for free!
- **Problems:**
 - is a guard true for a given valuation of the global variables?
 - does a guard have a solution? (SAT)
 - which solution to choose for a guard?
 - how can I reach a given location (symbolic reachability)?
 - what is a “useful” symbolic coverage criteria?
 - what is a symbolic test case?
 - ...

Symbolic Models

- **Possible solutions:**
 - choose a feasible subset of First-Order Logic
 - use techniques like constraint solving, model checking, theorem proving,...



A Library to Simulate STS

- A prototype Java library to simulate STS
- A subset of XML Schema data types is supported
- A simple language to express guards
- To solve guards, the constraint solver of GNU Prolog is used
- Limitations:
 - only positive integers are supported by the solver
 - other data types are mapped to integers:
 - simple types (boolean, enumeration, string)
 - complex types (to model classes)
 - some experimental features, like floats with fixed precision



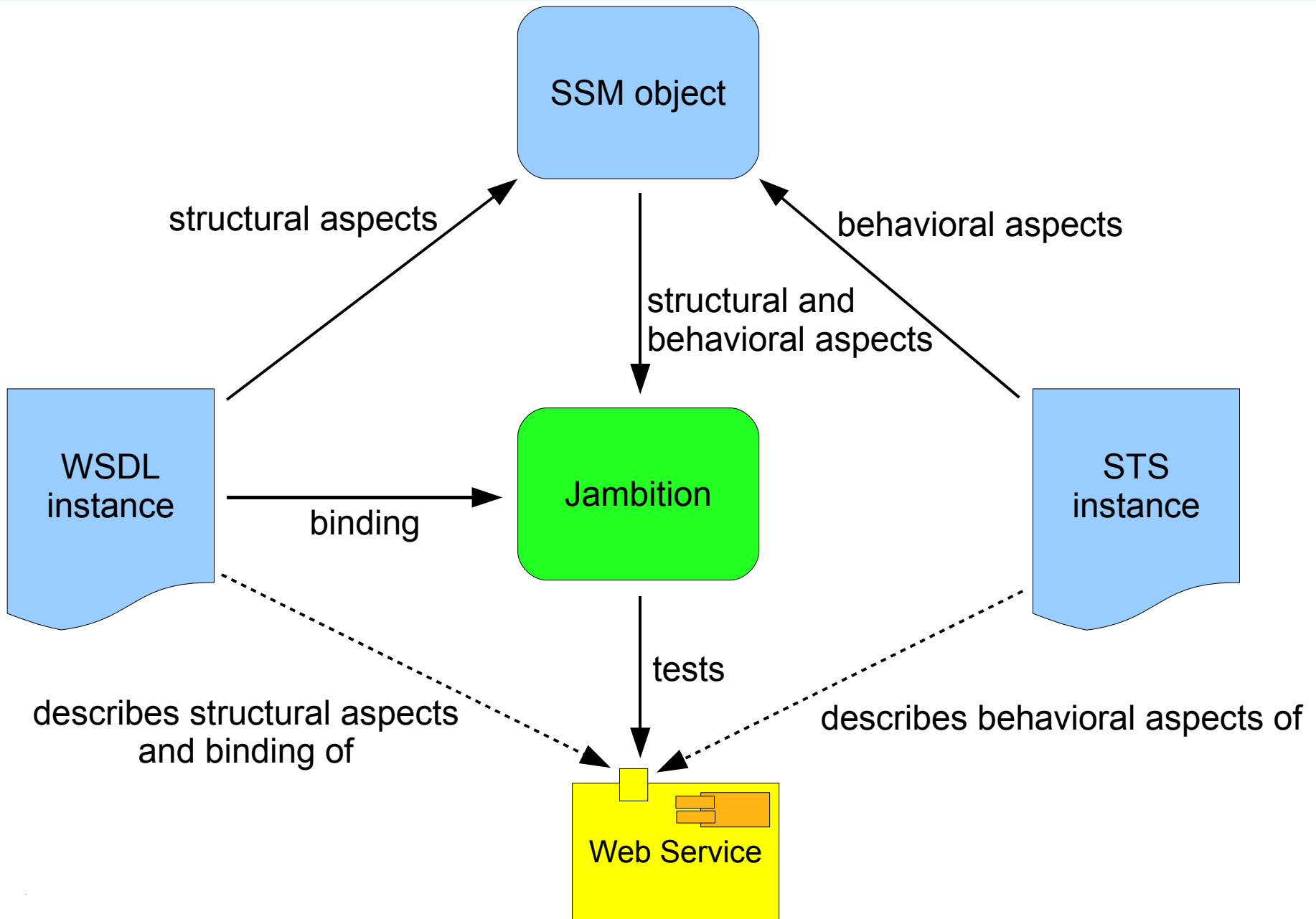
A Library to Simulate STS

- Using the library, on-the fly testing algorithms can be implemented, e.g. for:
 - ***sioco***
 - ***eco***
 - ?

Jambition

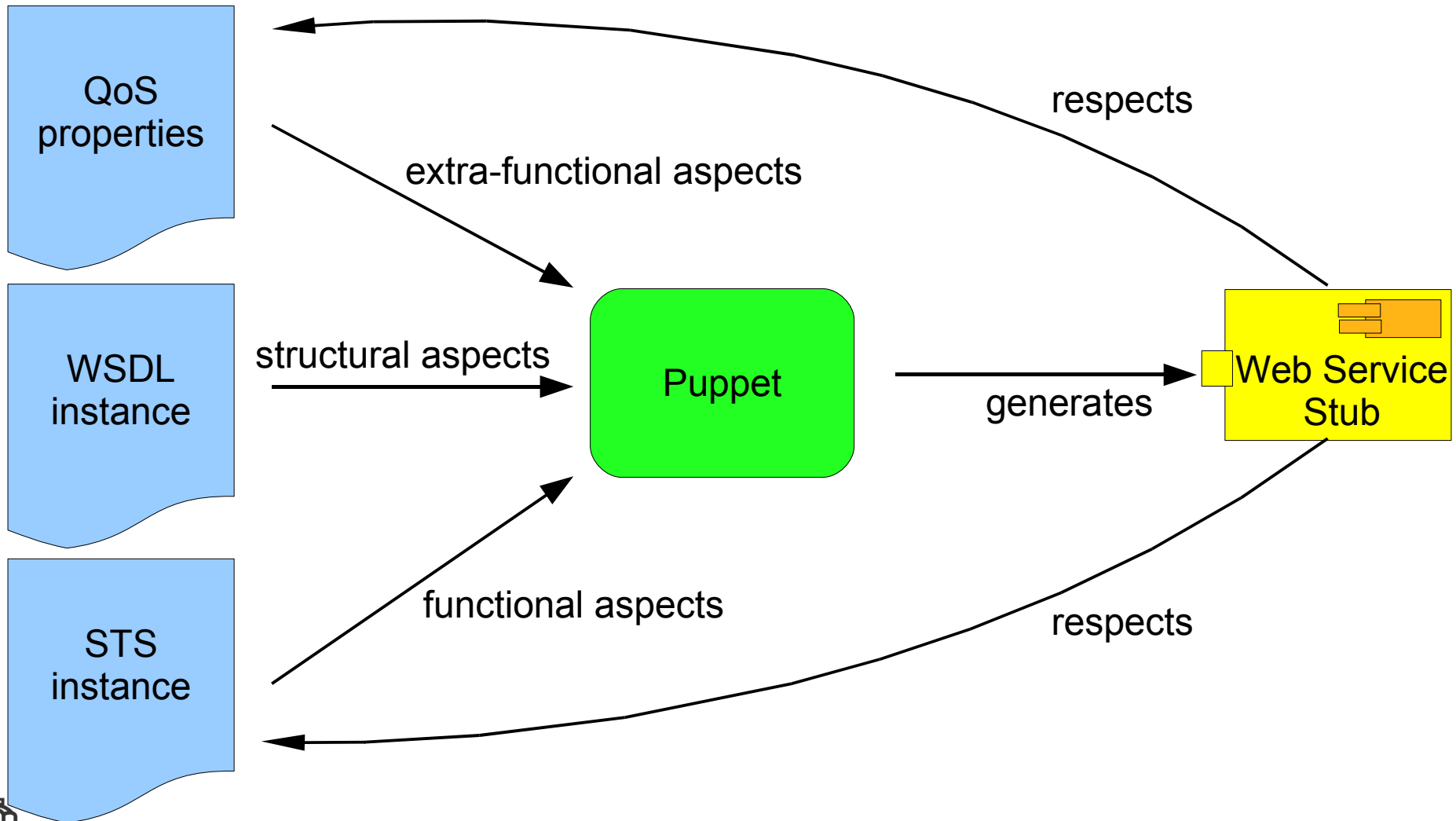
- A prototype tool for automatic testing of Web Services based on WSDL+STS specifications – **Jambition**
- Current status:
 - only passive Web Services
 - passive **sioco**
 - on-the-fly testing
 - random walk through the STS
- Jambition is part of the toolsuite of the EU **PLASTIC** project www.ist-plastic.org

Jambition



Puppet

- Goal: model-based generation of functional and extra-functional test-beds for Web Services



Audition

- Goal: create a directory service with trustable entities
- Approach: add a testing phase before a service gets registered
- Two levels of correctness:
 - correct provided functionality (passive **sioco**)
 - correct usage of environmental services (eco)
- As specifications serve STSs
- As testing engine serves a Web Service variant of Ambition

The End

- Thanks!

