Istituto di Scienza e Tecnologie dell'Informazione “A. Faedo”
Software Engineering Laboratory

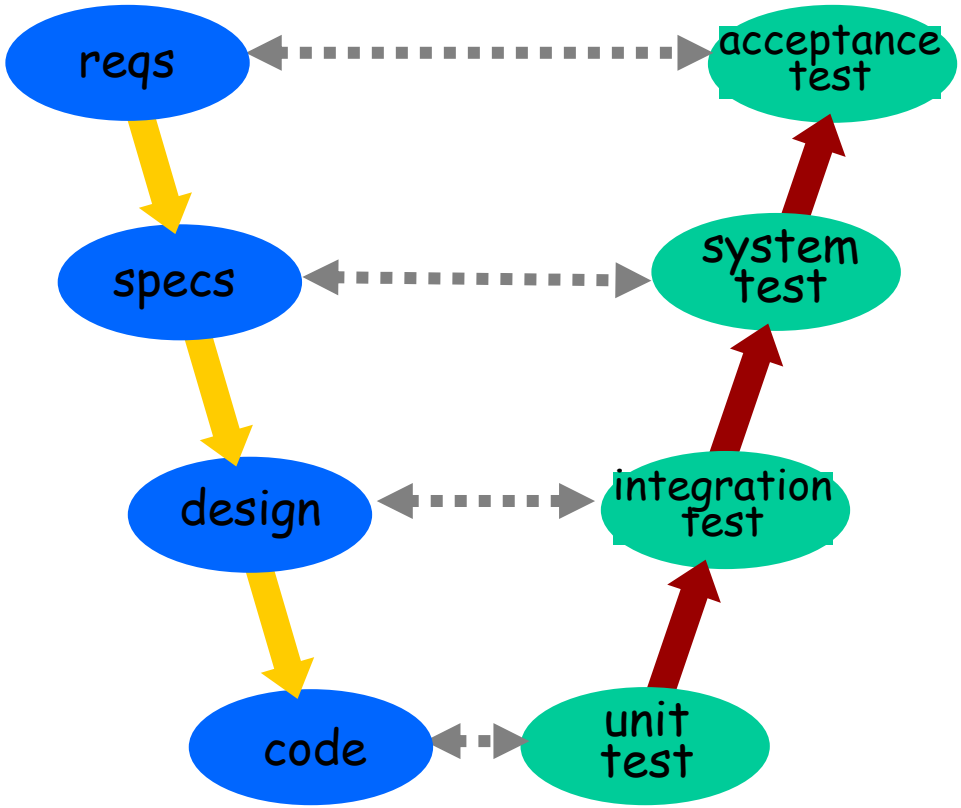
Methods and tools for the specification, verification and testing of functional and extra-functional properties of Dynamic Software Architectures

Attività su: *Tecniche e Strumenti per il Testing (WP4)*

Antonia Bertolino
antonia.bertolino@isti.cnr.it
http://www.isti.cnr.it/People/A.Bertolino

Classical V-Model

- Testing in the traditional lifecycle relies on planning ahead based on development artifacts, and testing before release:



The classical V-model presupposes that before a product is released, someone from the development organization or a third party is sufficiently acquainted with the intended system behavior and can control its configuration, so to select and launch an adequate executions sample (the test suite).

(Functional and non-functional) testing of DSAs

- DSA sono sistemi che si trasformano a run-time
- …a maggior ragione testing e monitoring sono necessari

- ..ma, challenge per testing: non si può fare V&V precoce..
- Impossibile prevedere mentre sviluppo un componente con chi e come interagira'..qdi servono approcci on-the-fly

- SOA è caso attuale e diffuso di DSA

Challenges in SOA testing

- Global protocols of service interactions
- Inter-organization distributedness
  - no access to code neither to its environment
  - no control over the service lifecycle
- Runtime discovery and binding
  - No sufficient pre-deployment testing strategy
- Runtime dynamism
  - binding and service implementation can inadvertently change at runtime
- Increased relevance of QoS characteristics
- Asynchronous and synchronous behavior combined
- Context-awareness

Context awareness

- A service modifies its behavior according to available services and support from the platform (the environment)

- Context can change at any instant
  - Software behavior in case of context change has to be specified
  - context evolution has to be reproduced for testing purpose

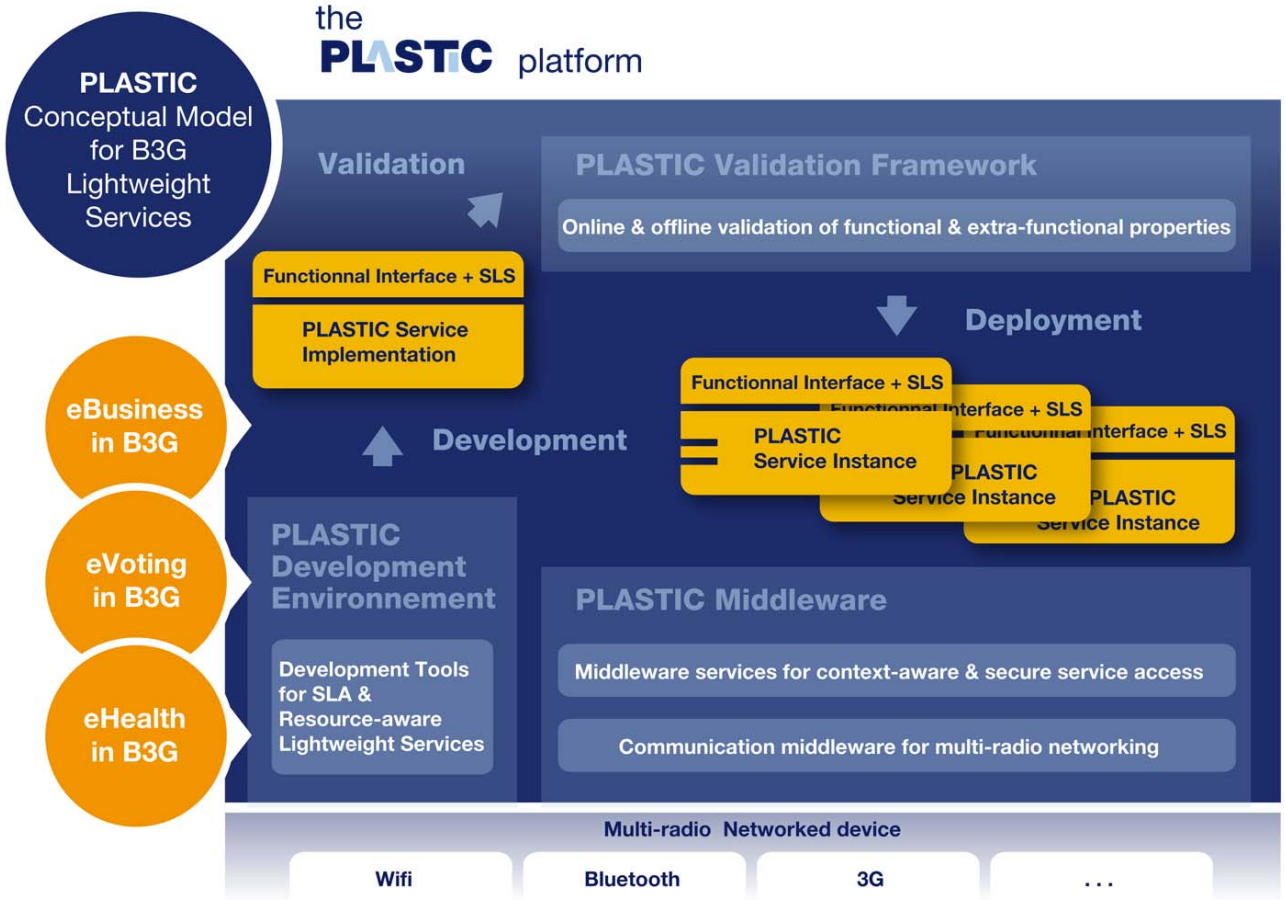
...and opportunities

- Declarative and formal specification publicly available both for functional and non functional characteristics
- New "control" point in the platform to which testing activities can be associated
  - E.g., directory services
- Open and standardized protocols and formats
  - SOAP, WSDL, XML
  - Orchestration provides a baseline for integration testing
  - Choreography provide an open specification to which cooperating services must conform
- Lack of trust among organization

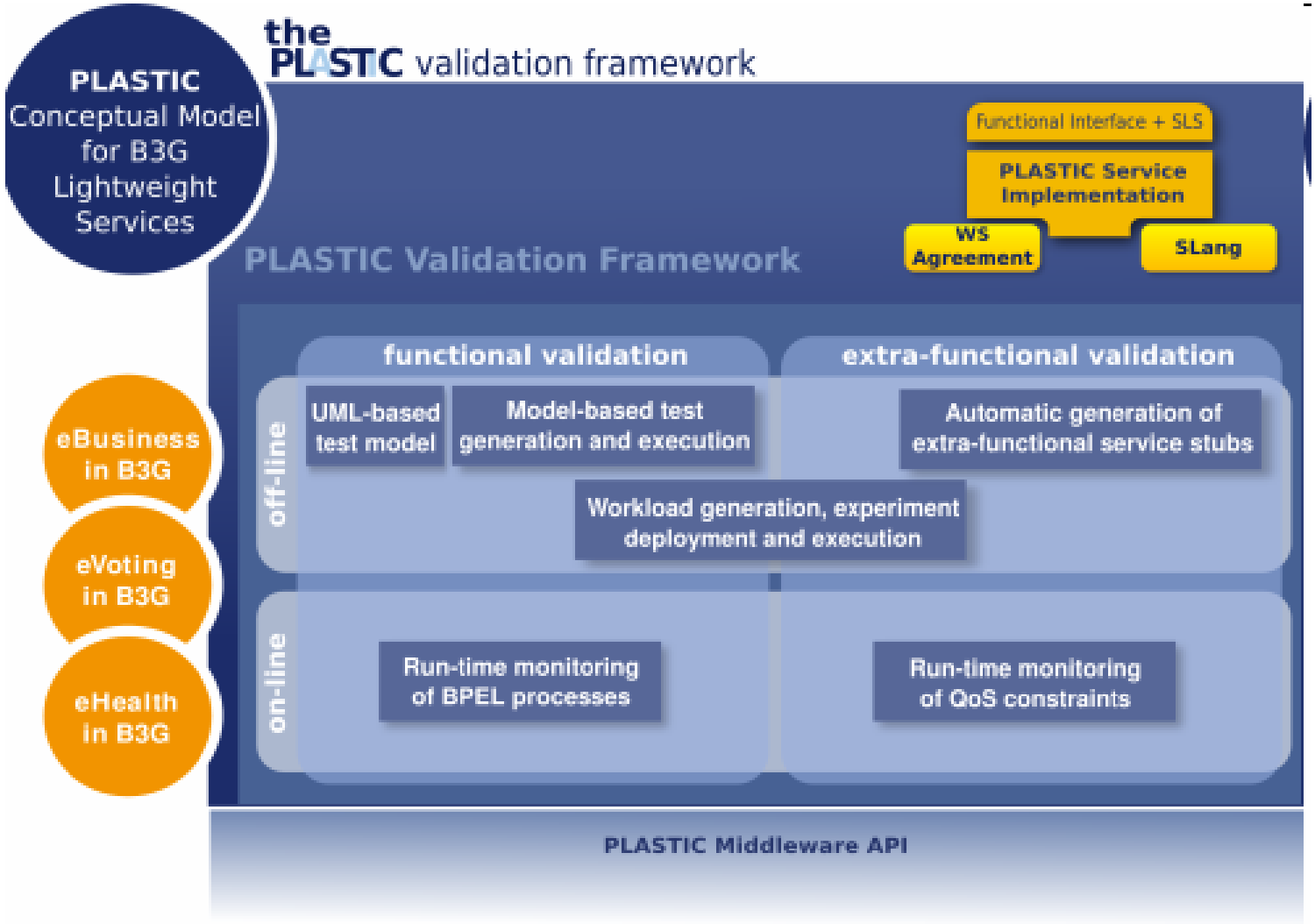
Progetti e attivita' in corso di rilievo per D-ASAP

- PLASTIC
- TAS3
- Art-Deco
- TAXI
- SOA Test Governance (Audition, SOCT)
- Connect

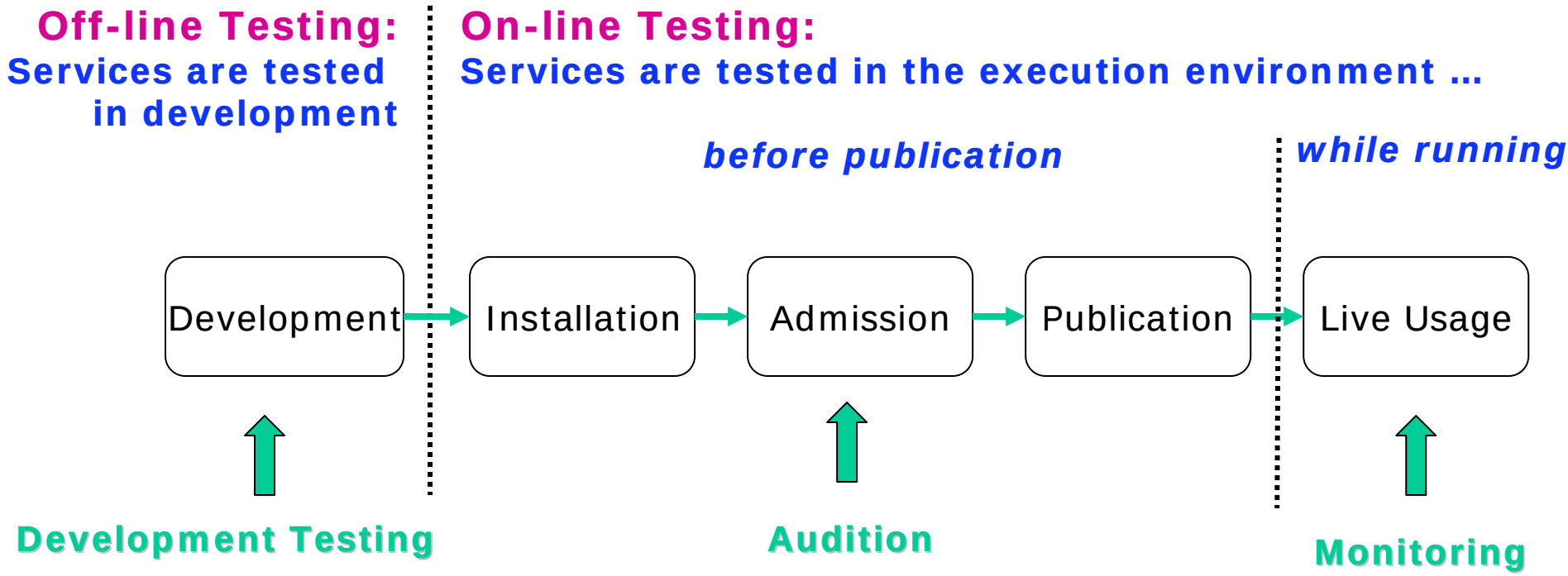
The European PLASTIC project aims at **P**roviding **L**ightweight and **A**daptable **S**ervice **T**echnology for Pervasive **I**nformation and **C**ommunication



PLASTIC Validation Framework



Three Testing Stages

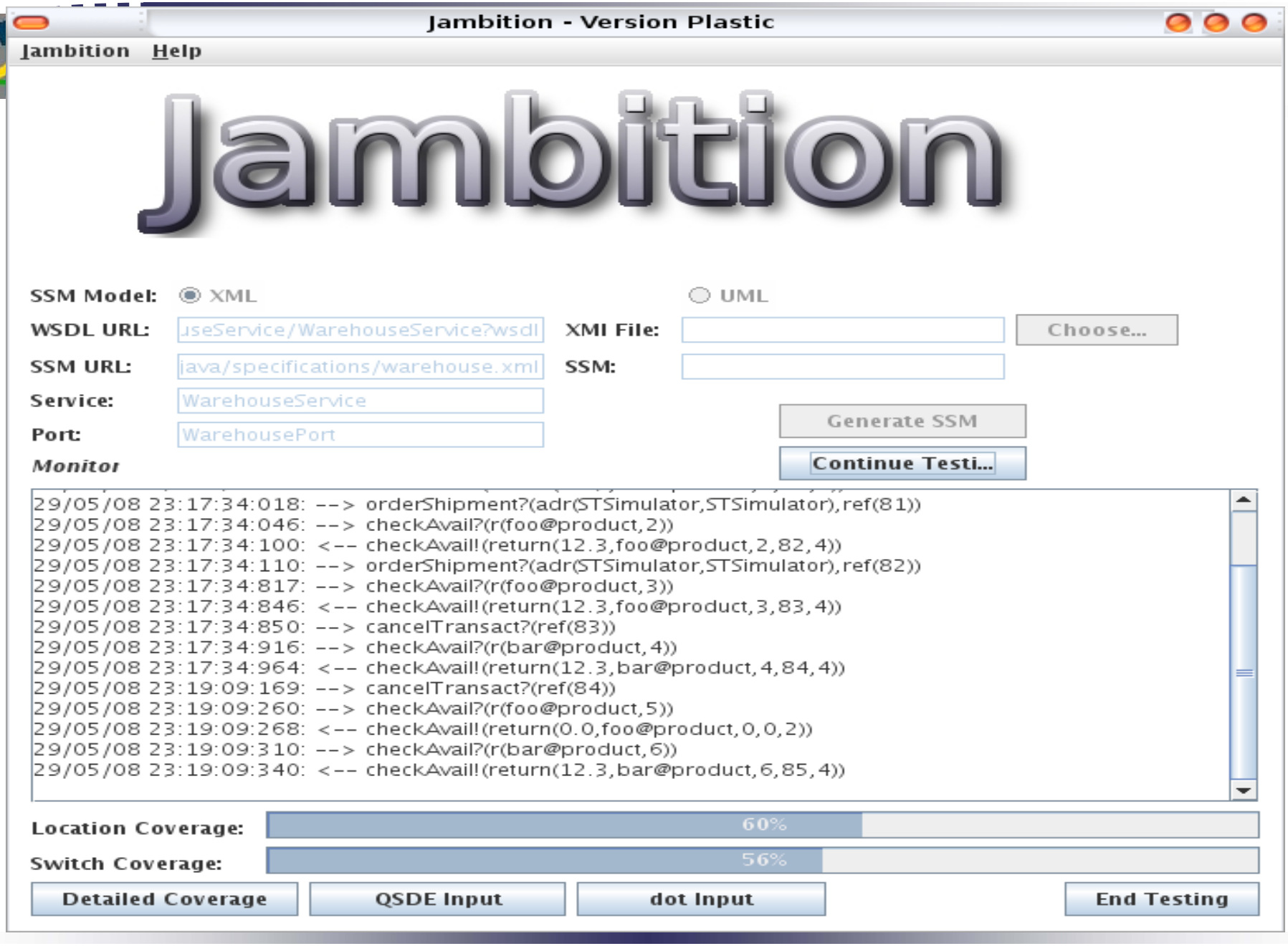


Plastic test tools repository:

http://plastic.isti.cnr.it

Tools developed in the PLASTIC Project

- Jambition tool for functional testing
- Puppet tool for QoS testbed
- Jambition+Puppet for Model-based QoS testing


Jambition - Version Plastic
Jambition Help
Jambition
SSM Model: XML UML
WSDL URL: useService/WarehouseService?wsdl
XMI File:
Choose...
SSM URL: java/specifications/warehouse.xml
SSM:
Service: WarehouseService
Port: WarehousePort
Generate SSM
Continue Testi...
Monitor
29/05/08 23:17:34:018: --> orderShipment?(adr(STSimulator,STSimulator),ref(81))
29/05/08 23:17:34:046: --> checkAvail?(r(foo@product,2))
29/05/08 23:17:34:100: <-- checkAvail!(return(12.3,foo@product,2,82,4))
29/05/08 23:17:34:110: --> orderShipment?(adr(STSimulator,STSimulator),ref(82))
29/05/08 23:17:34:817: --> checkAvail?(r(foo@product,3))
29/05/08 23:17:34:846: <-- checkAvail!(return(12.3,foo@product,3,83,4))
29/05/08 23:17:34:850: --> cancelTransact?(ref(83))
29/05/08 23:17:34:916: --> checkAvail?(r(bar@product,4))
29/05/08 23:17:34:964: <-- checkAvail!(return(12.3,bar@product,4,84,4))
29/05/08 23:19:09:169: --> cancelTransact?(ref(84))
29/05/08 23:19:09:260: --> checkAvail?(r(foo@product,5))
29/05/08 23:19:09:268: <-- checkAvail!(return(0.0,foo@product,0,0,2))
29/05/08 23:19:09:310: --> checkAvail?(r(bar@product,6))
29/05/08 23:19:09:340: <-- checkAvail!(return(12.3,bar@product,6,85,4))
Location Coverage:
60%
Switch Coverage:
56%
Detailed Coverage
QSDE Input
dot Input
End Testing

Warehouse Funct. Specs

- **Static** interface functionality, as being defined in the WSDL

Data

QuoteRequest
product: String quantity: Integer

Quote
status: Integer product: String quantity: Integer price: Double refNumber: Integer

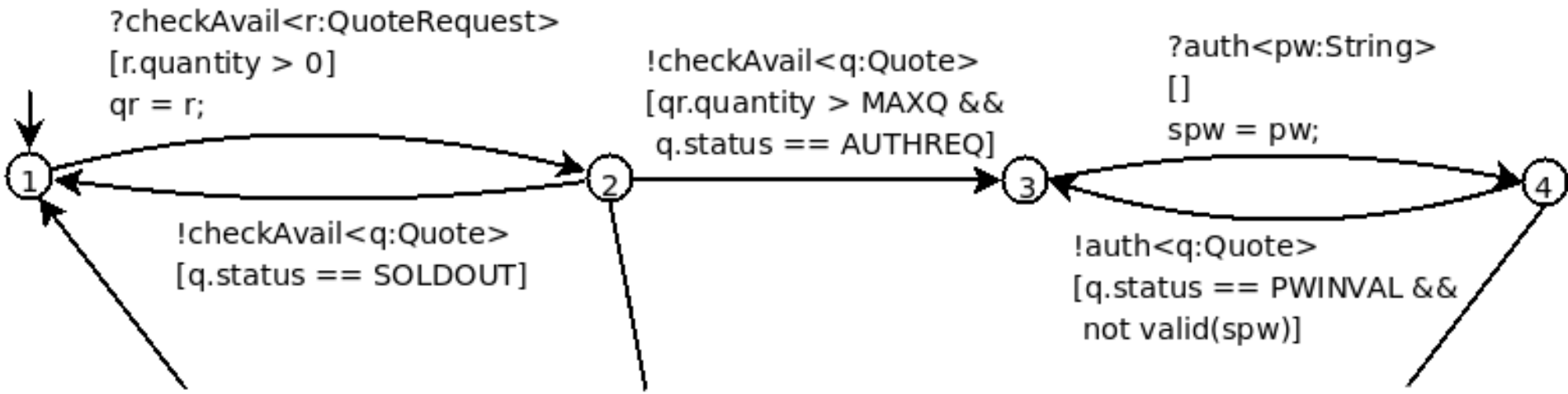
Address
firstName: String surname: String ...

Operations

Operation	Input Message	Output Message
checkAvail	?checkAvail(r : QuoteRequest)	!checkAvail(q : Quote)
auth	?auth(pw : String)	!auth(q : Quote)
cancelTransact	?cancelTransact(ref : Integer)	—
orderShipment	?orderShipment(ref : Integer, adr : Address)	—

- Many have proposed to enrich the WSDL with semantic information

Symbolic Transition Systems



- Introduce symbolic concepts like variables and guards over LTS simpler model
- STSs formally defined in:

  L. Frantzen, J. Tretmans, and T.A.C. Willemse. A Symbolic Framework for Model-Based Testing. Proc. FATES/RV 2006, LNCS 4262, 2006.

QoS testing

- Defining an agreement is not enough, we need to insert in the platform mechanisms to check if an agreement has been fulfilled or not

- Testing can be a useful mean to assess QoS by
  - reproducing different usage scenarios and deploying the service on a testbed
  - relevant information to be used at runtime during contract definition can be observed

- Services in general may invoke other services in order to carry out the computation requested by the clients.
  - If this invocation is directed to a service that does not refer to stateful resources, then for testing purposes it is possible to use existing and already running services.

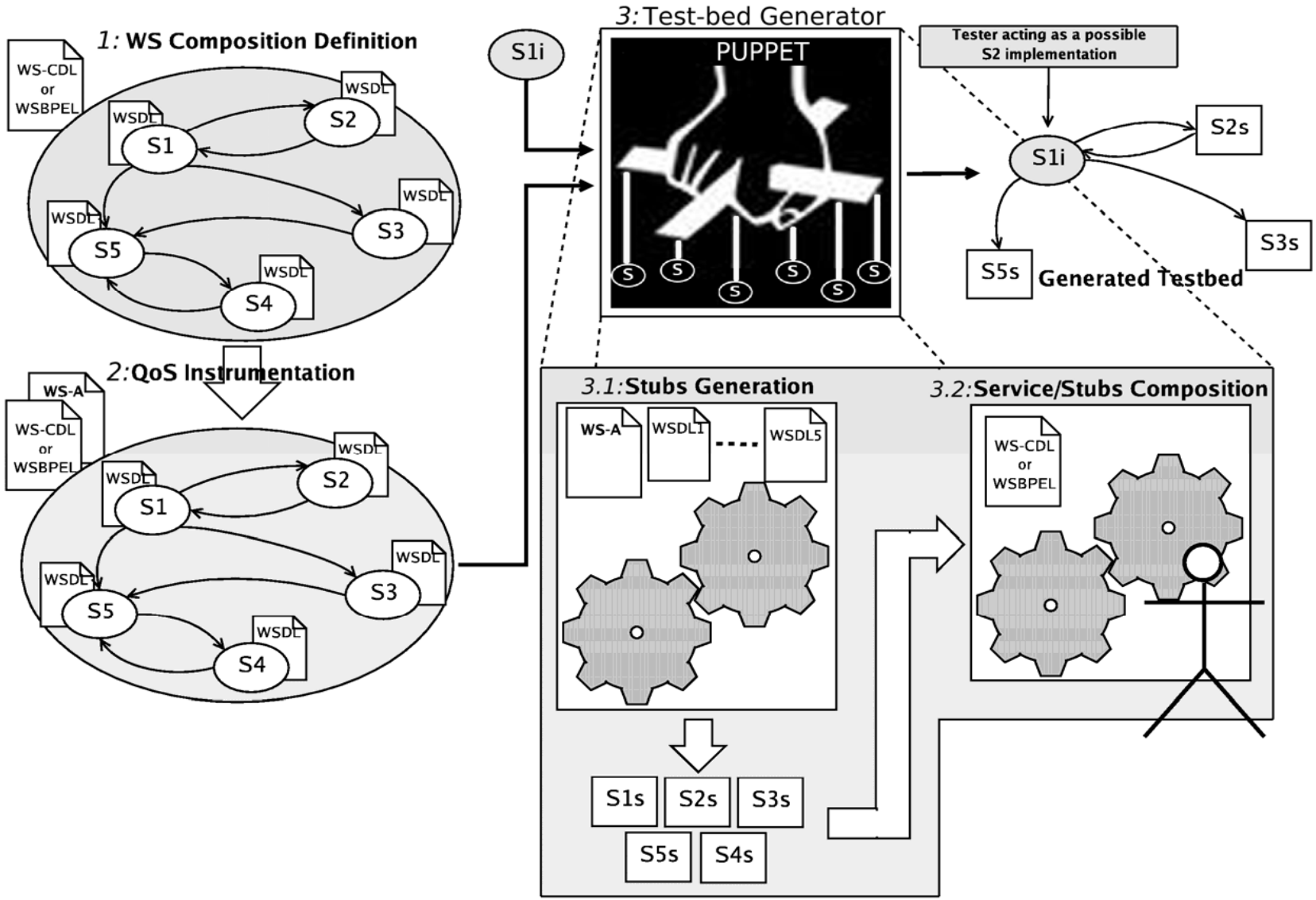
  - Conversely, if the invoked service accesses stateful resources, this option must be ruled out and the required services have to be simulated.

PUPPET

- PUPPET is a framework meant to recreate a service's deployment environment
- The recreated environment becomes the testbed in which the service under development can be deployed in order to assess its QoS properties
- The main issue is then to recreate a trustable testbed

- A. Bertolino, G. De Angelis, A. Polini "A QoS Test-bed Generator for Web Services" in Proceedings of ICWE 2007, Como, Italy, July 16-20
- A. Bertolino, G. De Angelis, F. Lonetti, A. Sabetta "Let The Puppets Move! Automated Testbed Generation for Service-oriented Mobile Applications", in Proc. of 34th EUROMICRO Conf. on Sw Eng. and Advanced Applications (SEAA 2008) , Sept. 3-5, Parma, Italy, IEEE CS, 2008, pp. 11-19 .

Puppet steps



PUPPET + Jambition

1. Convert WSDL to Java Service skeleton (WSDL2Java tool)

2. Extend with extra-functional code

3. Extend with functional code (via embedding the STSimulator simulating ***eco***)

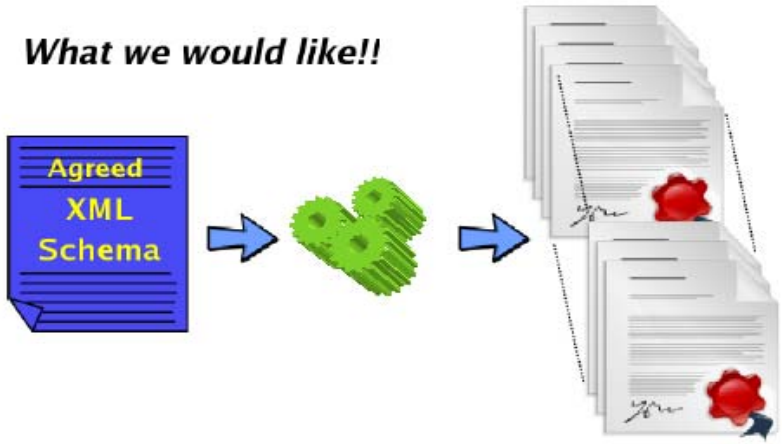
A. Bertolino, G. De Angelis, L. Frantzen, A. Polini "Model-Based Generation of Testbeds for Web Services", in Proc. of Testcom/Fates 2008, June 10-13,Tokyo, Japan, LNCS 5047, 2008, pp.266-282.

XML-based testing

- The availability of XML schemas describing the structure of the messages exchanged by services can be exploited to generate compliant messages to apply data analysis testing approaches
- TAXI is a tool for systematic document generation from XML Schema
- WS testing based on TAXI: on-going work

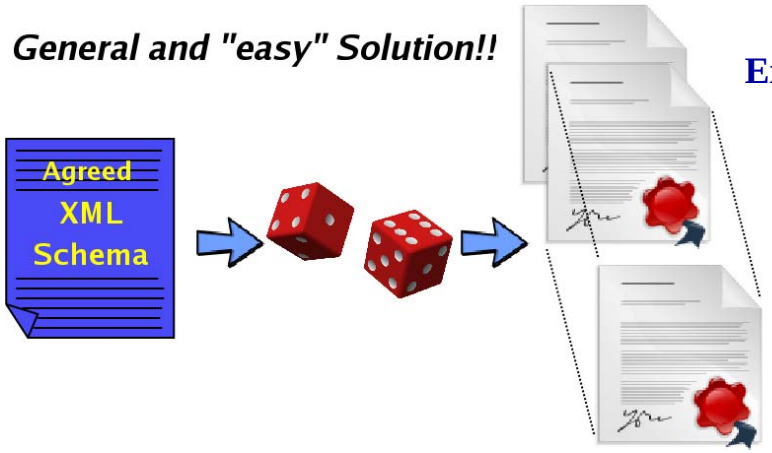
Automatic XML-Based Testing

What we would like!!



Generally not Finite set!!

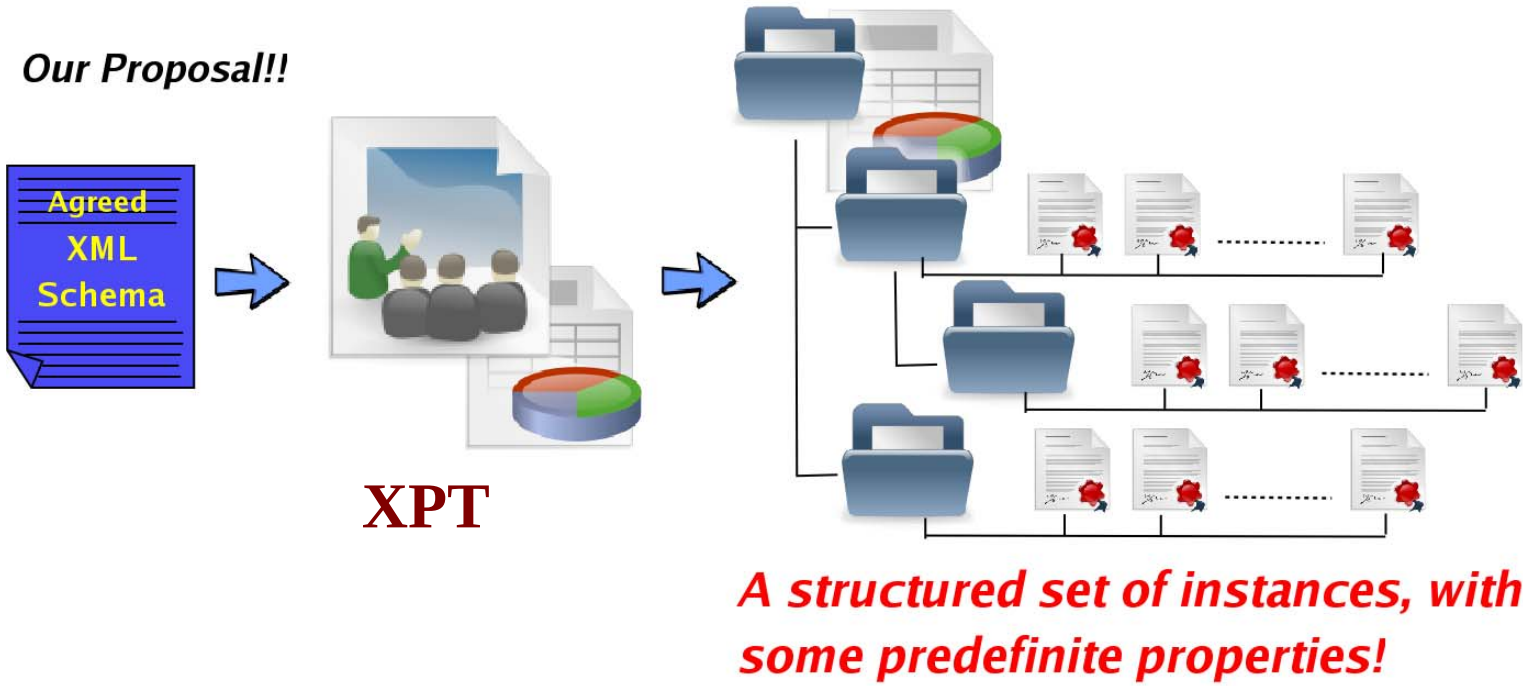
General and "easy" Solution!!



Existing Tool: XMLSpy, sunXMLGenerator

Not Structured set, difficult to derive properties!!

A Systematic Approach

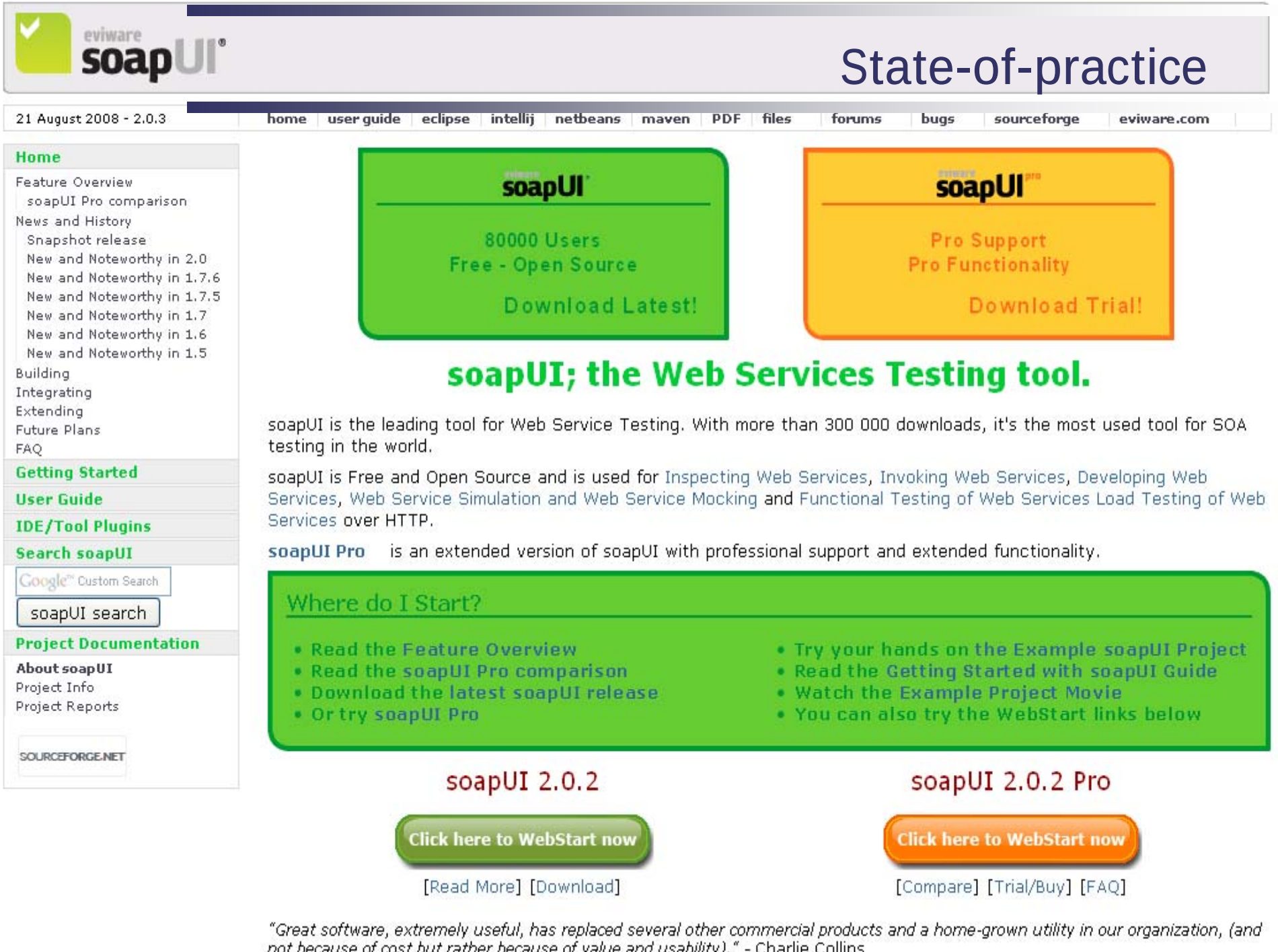


The approach is based on the well-known Category Partition methodology

Potential Applications

- For validating database management systems
  - automatically generate valid XML instances for populating database in a systematic
  - evaluate the performance and the quality of the associated management systems
- For testing the inter-operability between applications and for enabling the correct interactions among the interfaces used by remote components in distributed systems.
  - Automatic and controlled generation of valid and invalid instances enables the automated testing of I/O behavior
- For verifying the proper communication protocols between web-services.
  - SOAP-based interaction between services can be reconducted to the corresponding XML Schemas

State-of-practice



soapUI limitations

- question marks (or alternatively some pregenerated Latin words or random numbers) in the place of actual input data for the generated test cases
- it does not manage the choice and all elements and the occurrence attributes, but introduces comments explaining to the human tester how to handle them
- the tool distinguishes the basic data types but with little variability
- *Our proposal is to overcome soapUI limitations by automatically handling above issues*

WS-TAXI

- The TAXI tool has been extended to generate a set of correct messages and incorrect SOAP messages for checking how an invoked WS reacts

- An open issue with syntax-based approaches is the automatic derivation of an "oracle" for the defined messages

  Work to be presented at ICST 2009

- Definition of Coverage for Def-use on BPEL

- C. Bartolini, A. Bertolino, E. Marchetti, I. Parissis. "Data Flow-based Validation of Web Services Compositions: Perspectives and Examples" , in Architecting Dependable Systems V, R. de Lemos, F. Di Giandomenico, H. Muccini, C. Gacek, M. Vieira (Eds.), LNCS 5135, 2008, pages 298,325.

SOA Test Governance

- Idea di base: testing si puo' effettuare solo grazie a stabilire convenzione fra stakeholders

- Esempi: Audition, SOCT

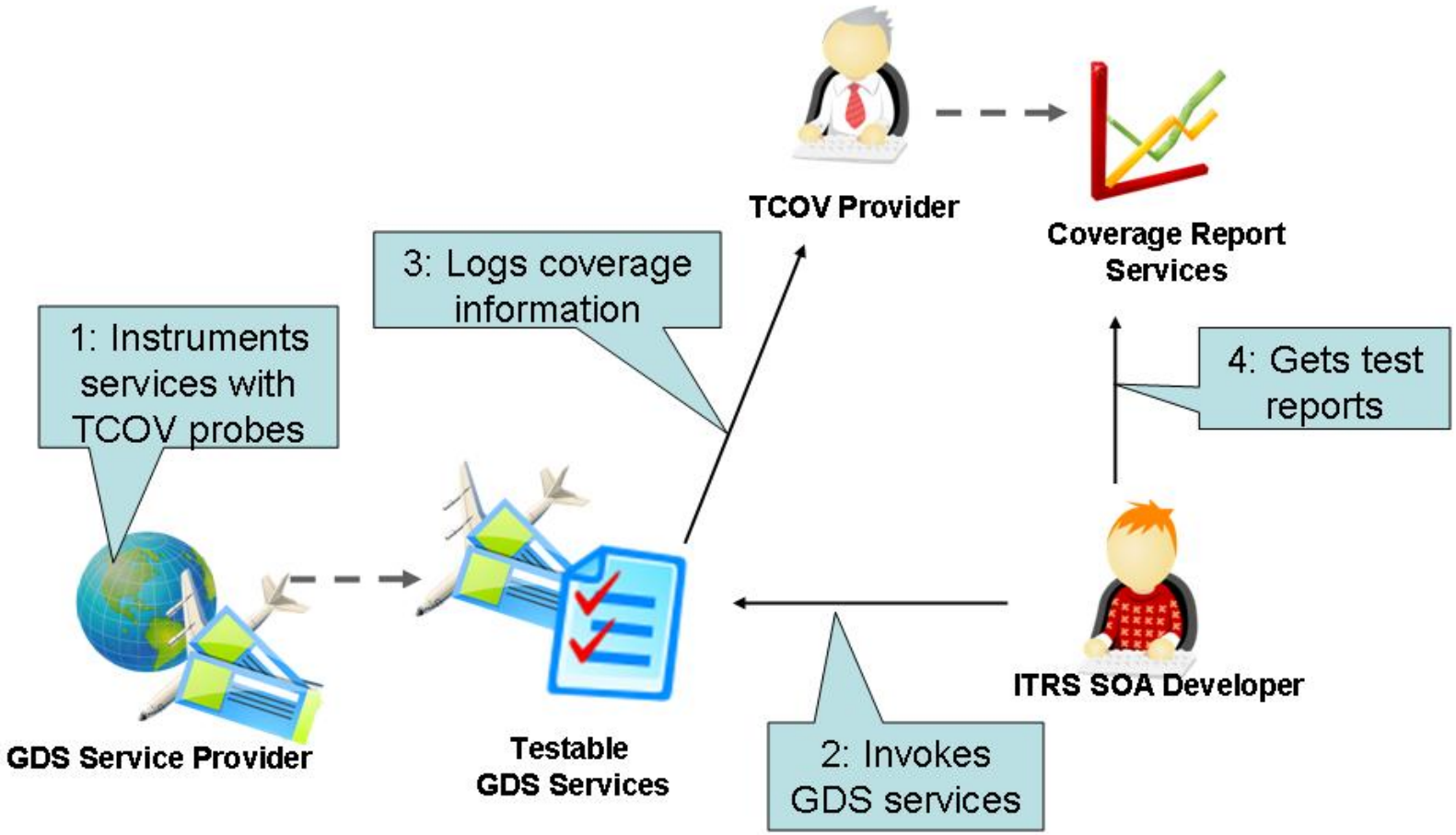
Audition testing

- The Audition testing approach tries to **add a testing phase before the service goes on the scene**

- The objective is to create a discovery service with
- **trustable entries** registration is guaranteed only to services carefully tested

- Development and study carried on for a registry based approach
  - **WSGuard** (Guaranteeing UDDI Audition at Registration and Discovery)

SOCT idea


TCOV Provider
Coverage Report Services
3: Logs coverage information
1: Instruments services with TCOV probes
4: Gets test reports
GDS Service Provider
Testable GDS Services
ITRS SOA Developer
2: Invokes GDS services

Conclusions

- Puppet, (WS-)TAXI ready for applications: looking for case studies
- Audition/SOCT under development
- Also starting approaches to monitoring of DSA within Connect

  - Thanks/Questions