Formal Modeling and Discrete-Time Analysis of BPEL Web Services

EOMAS 2008

June 16-17, 2008

Sylvain Rampacek
Sylvain.Rampacek@u-bourgogne.fr
LE2I (UMR CNRS 5158) - A5

Radu Mateescu
Radu.Mateescu@inria.fr
INRIA - VASY

UNIVERSITÉ DE BOURGOGNE

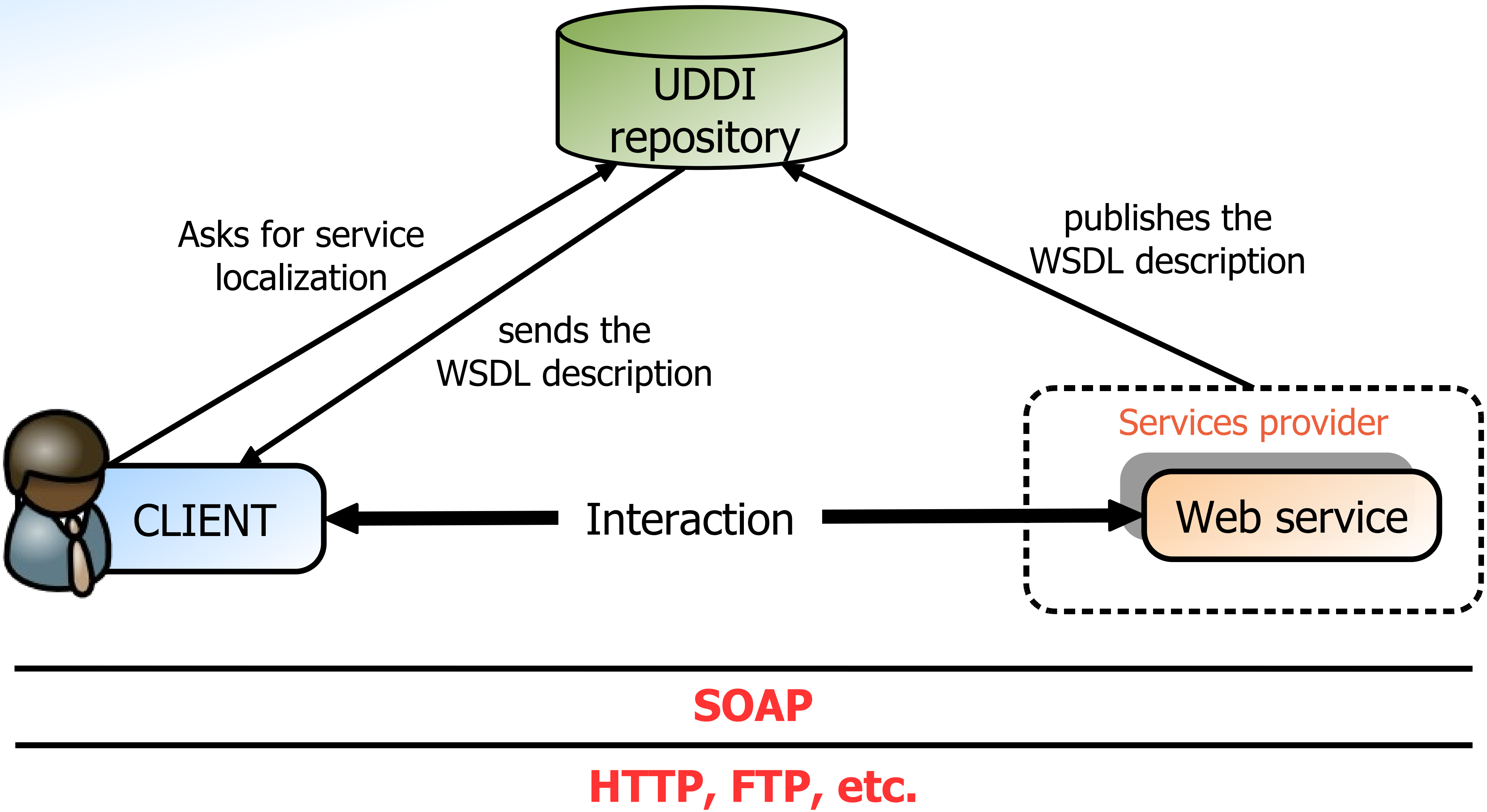
Laboratoire Electronique Informatique et Image

Summary

- **1. Introduction**
- **2. Modeling and Analysis Approach**
  - Translation from BPEL to discrete-time LTS
  - Analysis of discrete-time LTSs
- **3. Case Study: A Web Service for GPS Navigation**
- **4. Discrete-time LTS analysis**
- **5. Conclusion**

1. Introduction

Web services architecture



Web services: protocols and languages

XML technologies

- **Communication protocol: SOAP**
  - describe intermediary nodes (routing)
  - transport data (using XML)
- **Description language: WSDL**
  - describe the interface of the service
  - including types, parameters and methods
- **Behavioral description language: BPEL**
  - describe business-process
  - show articulation between each actions (including timing constraints)

Motivation

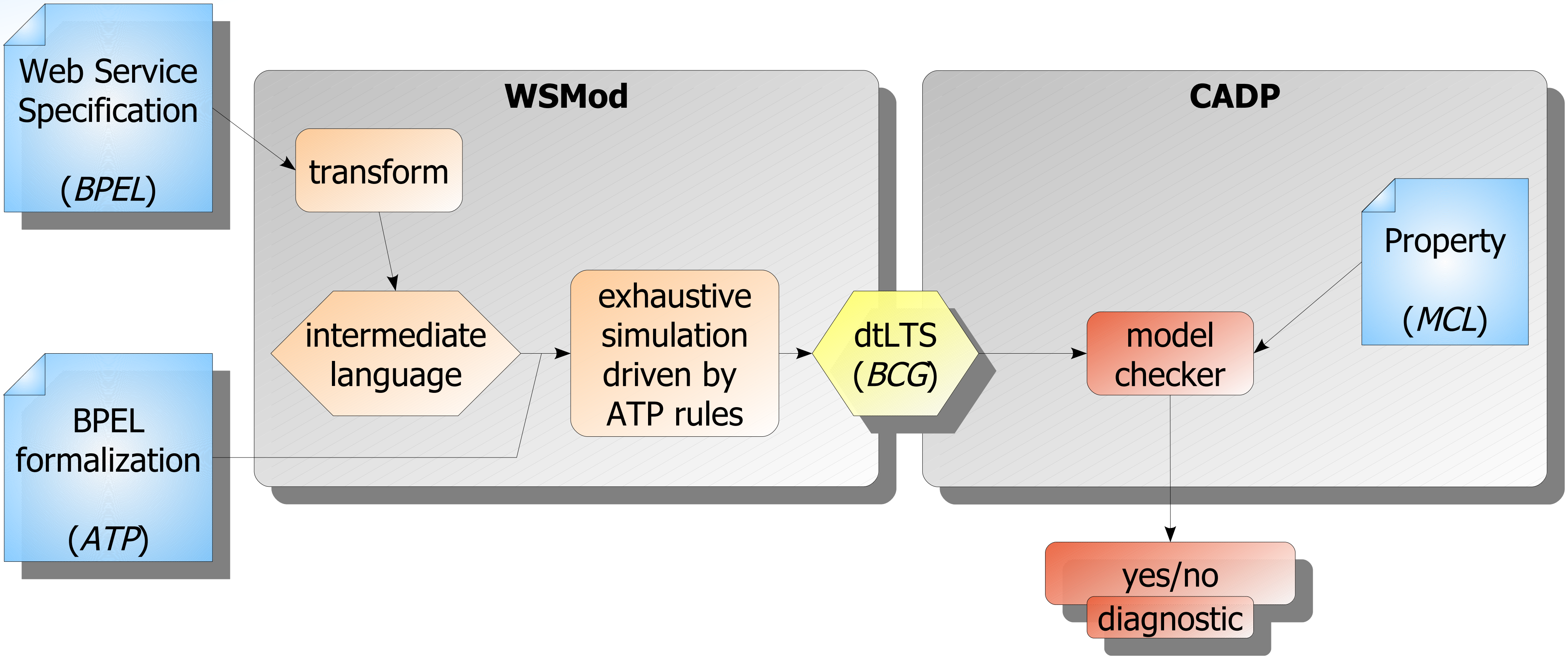
Analysis of Web services

- **Formal modeling and analysis of business processes**
  - Web services described in BPEL
- **Our approach** :
  - Defined a **formal semantics** of BPEL terms
    - *algebraic rules, taking into account the discrete-timing aspects*
  - **Automated generation** of models from BPEL specifications
    - *using an exhaustive simulation based on the formal semantics rules (using WSMod tool)*
  - **Analysis of resulting models** by using verification tools for concurrent systems (CADP)

2. Modeling and Analysis Approach

Platform for Web service modeling and analysis

Approach



2.1 Translation from BPEL to discrete-time LTS

Behavior of a Web service

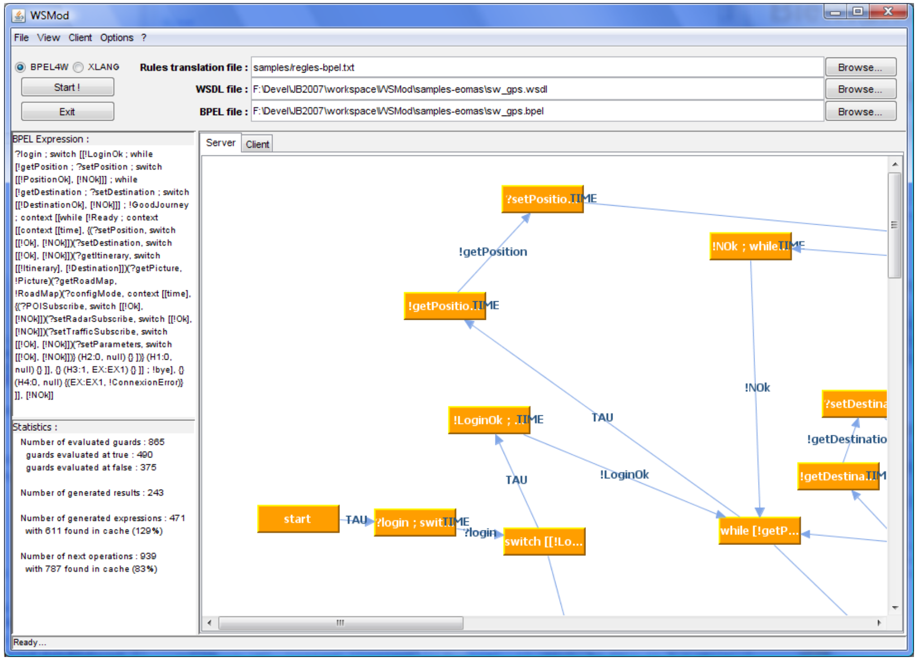
Actions and Processes in our formal semantics

- **Elementary actions**
  - send/receive (!/?)
  - internal action (τ), terminating action ($\surd$)
  - elapsing time (χ)
- **Elementary processes**
  - elementary action associated
- **Structured processes**
  - such as *sequences* of actions/processes, *loops*, *choices*, ...
- **Advanced processes**
  - such as *guarded execution by events or time*, *parallel execution*

Behavior of a Web service

Obtained by WSMod tool

- **WSMod tool:**
  - exhaustive simulation of BPEL description
  - discrete and continuous time representations
- **Two inputs:**
  - a Web service description in BPEL
  - a formal representation of BPEL semantics: based on Algebra of Timed Processes (ATP)
- **Results:**
  - dtLTS (discrete time)
  - timed automaton (dense time)



Timed semantics of BPEL process

"elementary" processes (extracts)

"elementary" processes

operation process

$$*o[m] \xrightarrow{*m} empty$$

$$*o[m] \xrightarrow{\chi} *o[m]$$

empty process

$$empty \xrightarrow{\surd} 0$$

throw process

$$throw[e] \xrightarrow{e} 0$$

time process

$$time \xrightarrow{\chi} time$$

Discrete *and* ***continuous time*** *rules*

Only ***discrete time*** rules

Timed semantics of BPEL process

"structured" processes (extracts)

***Discrete** and **continuous time** rules*

sequence process *(P;Q)*

$$\forall a \neq \surd \quad \frac{P \xrightarrow{a} P'}{P;Q \xrightarrow{a} P';Q}$$

$$\forall a \quad \frac{P \xrightarrow{\surd} \quad \wedge \quad Q \xrightarrow{a} Q'}{P;Q \xrightarrow{a} Q'}$$

switch process

$$switch[\{P_i\}] \xrightarrow{\tau} P_i$$

while process

$$while[P] \xrightarrow{\tau} empty$$

$$while[P] \xrightarrow{\tau} P;while[P]$$

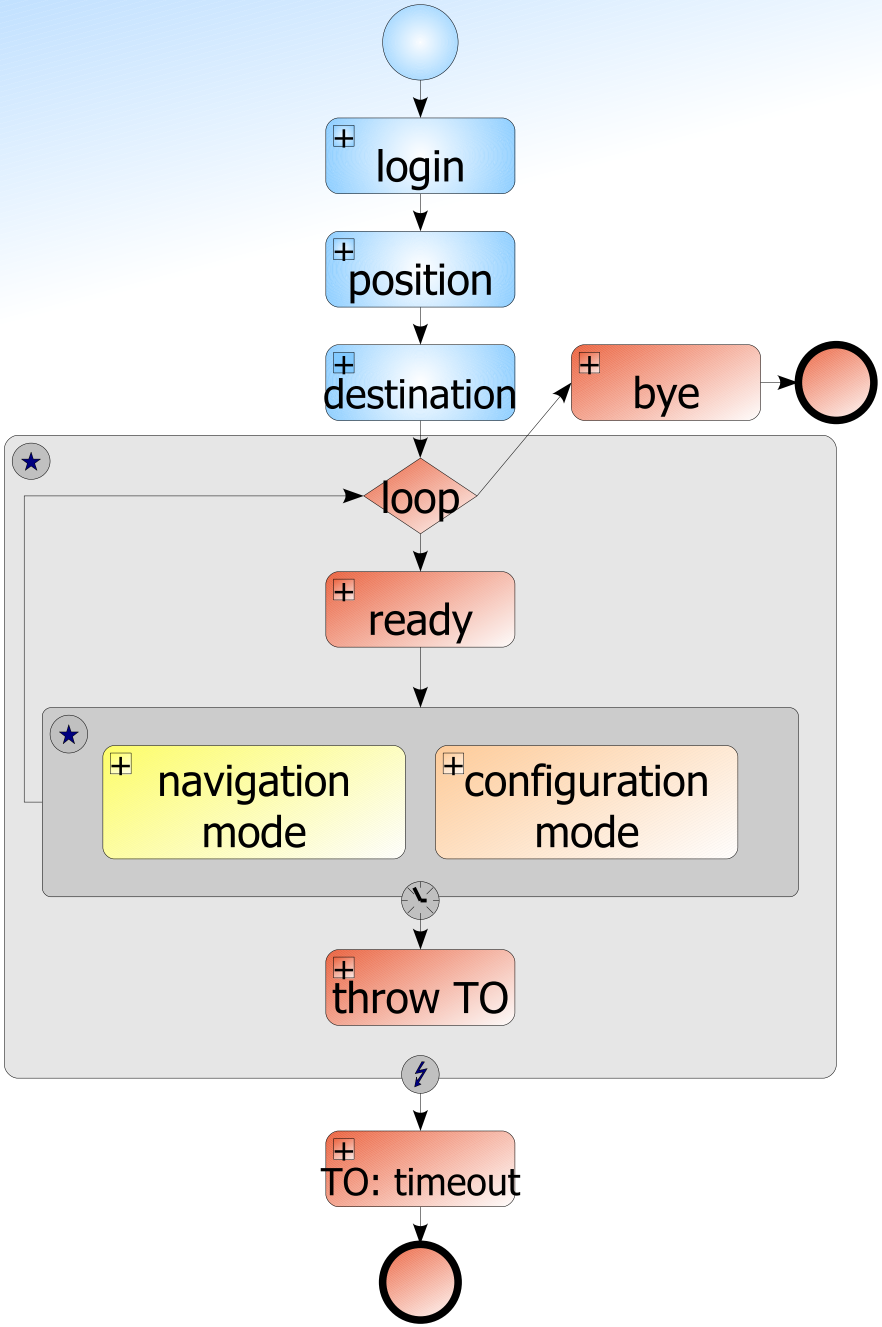
2.2 Analysis of discrete-time LTSs

CADP toolbox

- **Developed by INRIA-VASY team**
- **Contains currently over 40 tools and libraries for LTS manipulation**
  - batch mode: SVL, interactive mode: EUCALYPTUS
- **An LTS can be represented by two methods:**
  - explicitly: a list of states and transitions encoded into a file in the BCG format
  - implicitly: a successor function given as a C program, complying to the interface defined by Open/Caesar
- **In the sequel, we use the model checker Evaluator 4**
  - evaluate discrete-time properties on the dtLTS

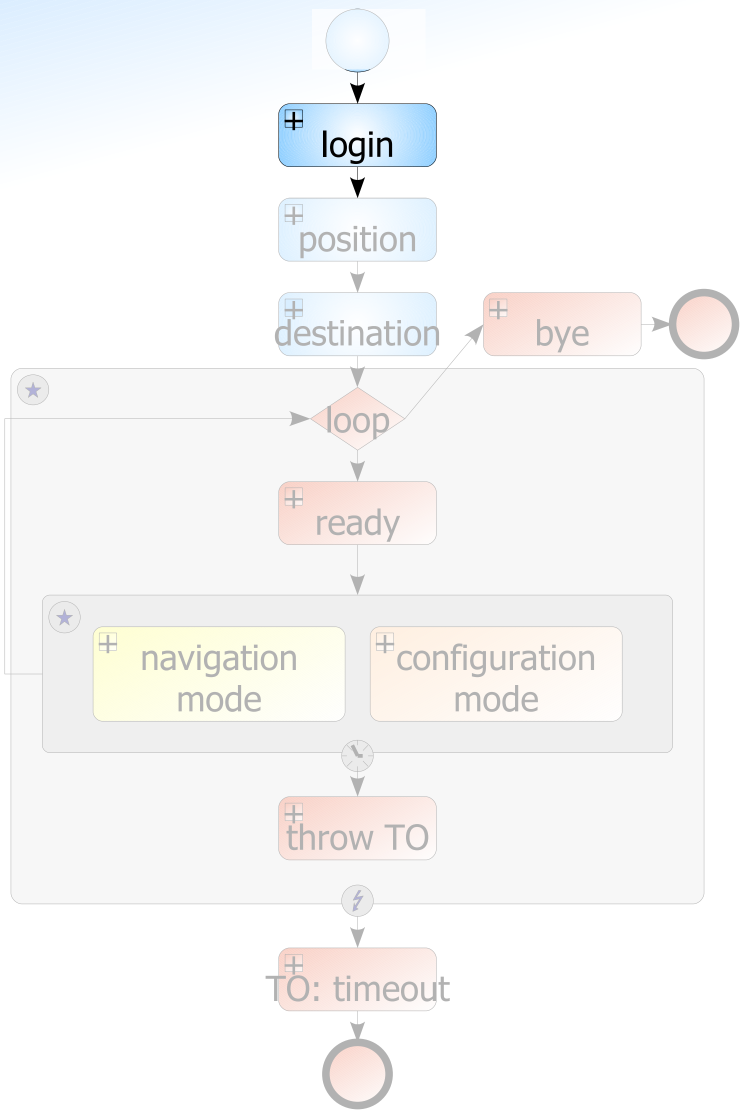
3. Case Study:
A Web Service for GPS Navigation

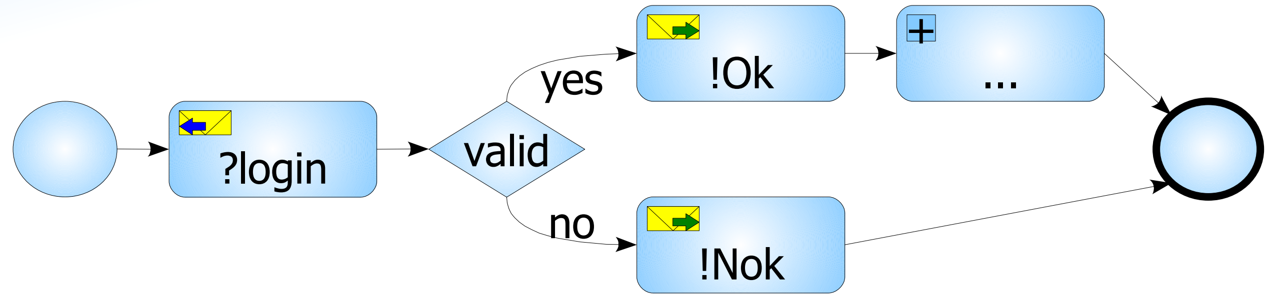
A Web Service for GPS Navigation



- **An initialization step**
  - login
  - get position
  - get destination
- **A main loop**
  - navigation mode
  - configuration mode

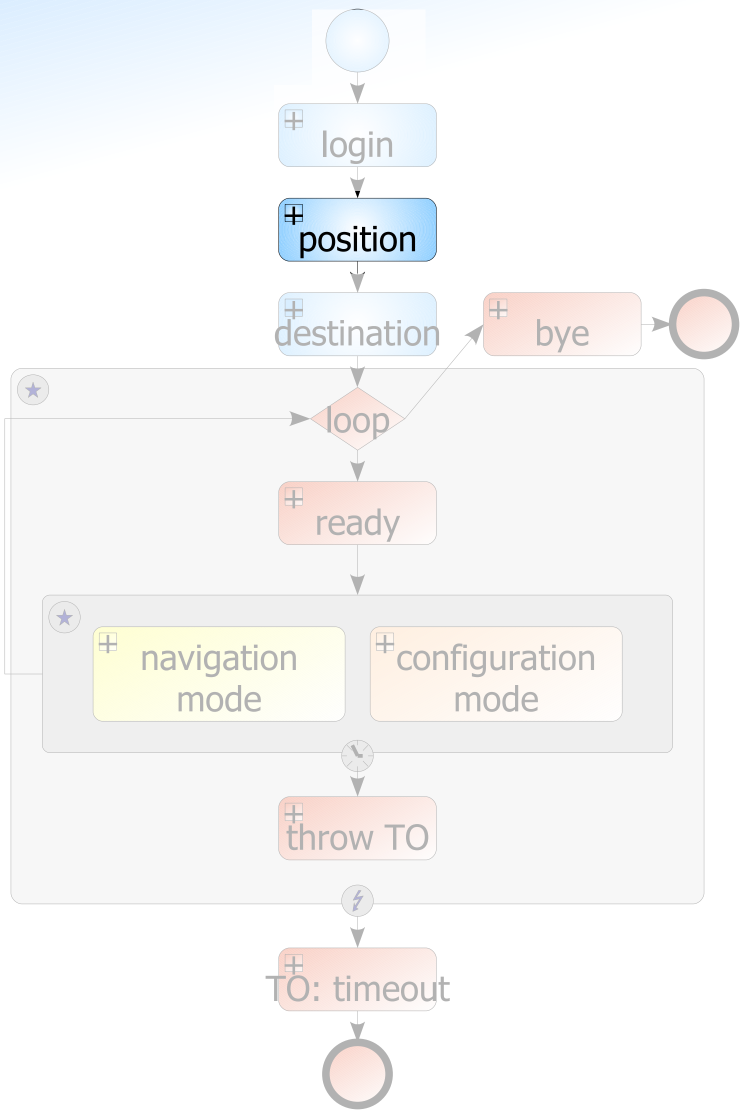
Zoom: login activity

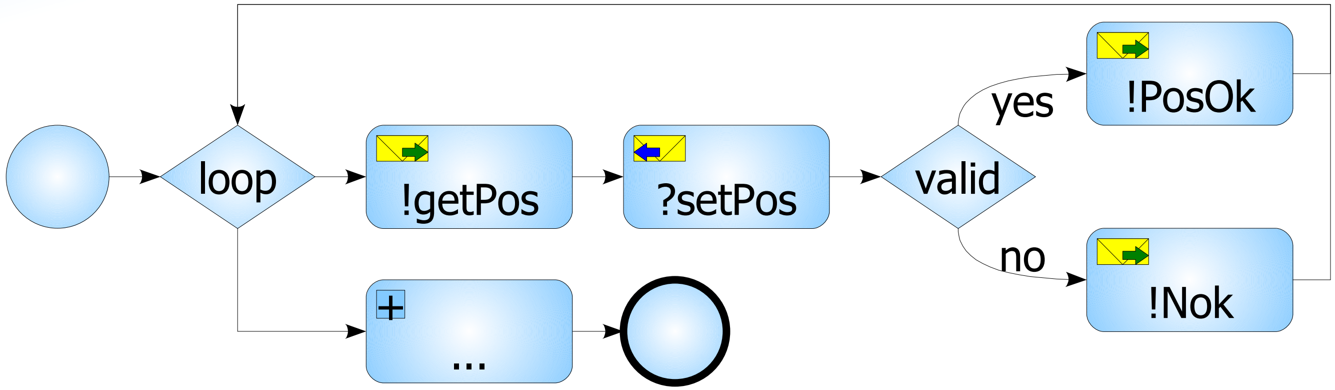




- **Message login (receive by the service)**
  - Containing login + password
- **The service returns *Ok* or *Nok***
  - Depending whether the couple is valid or not

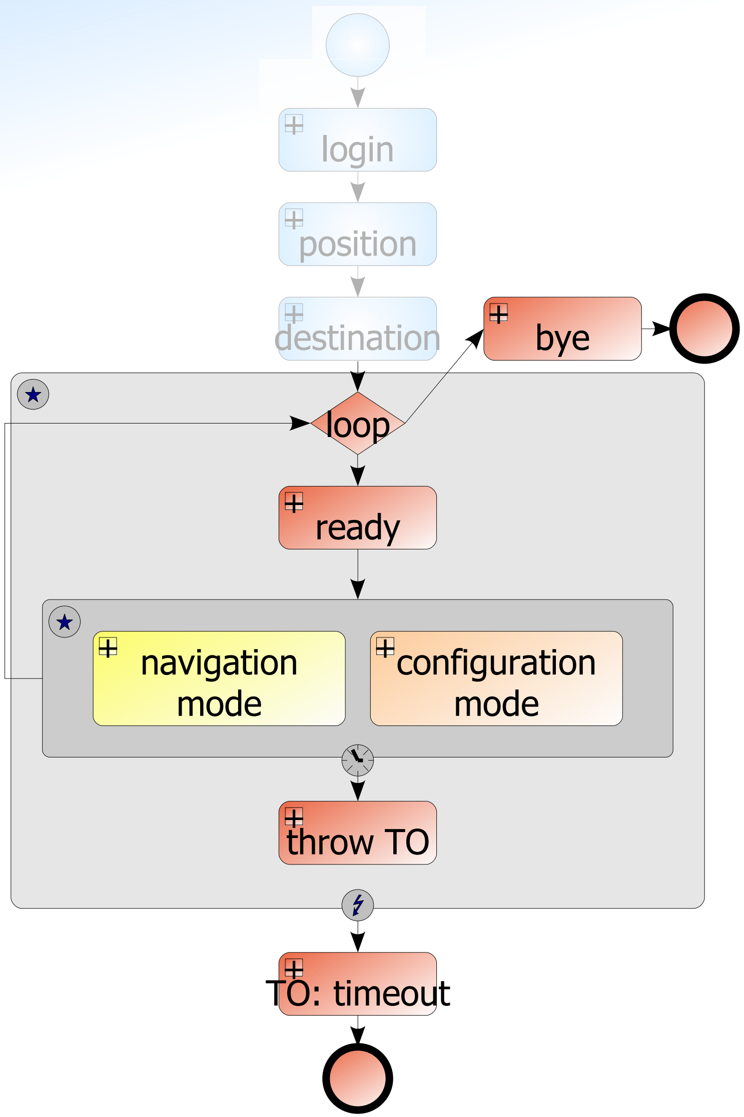
Zoom: position activity





- **The user must indicate its current position**
- **The service returns *PosOk* or *Nok***
  - Depending whether the position is valid or not
- **Same for destination activity**

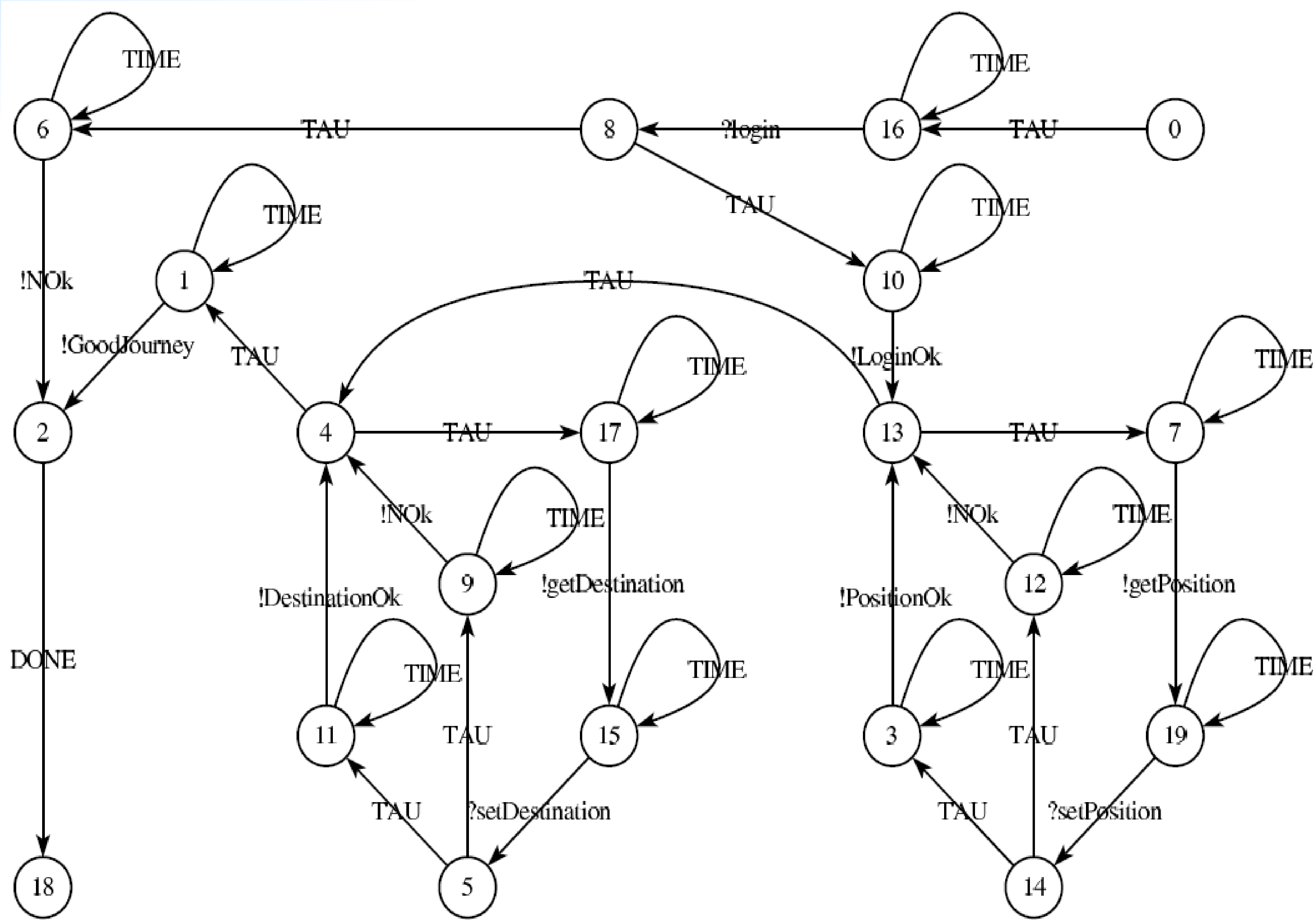
Zoom: main loop



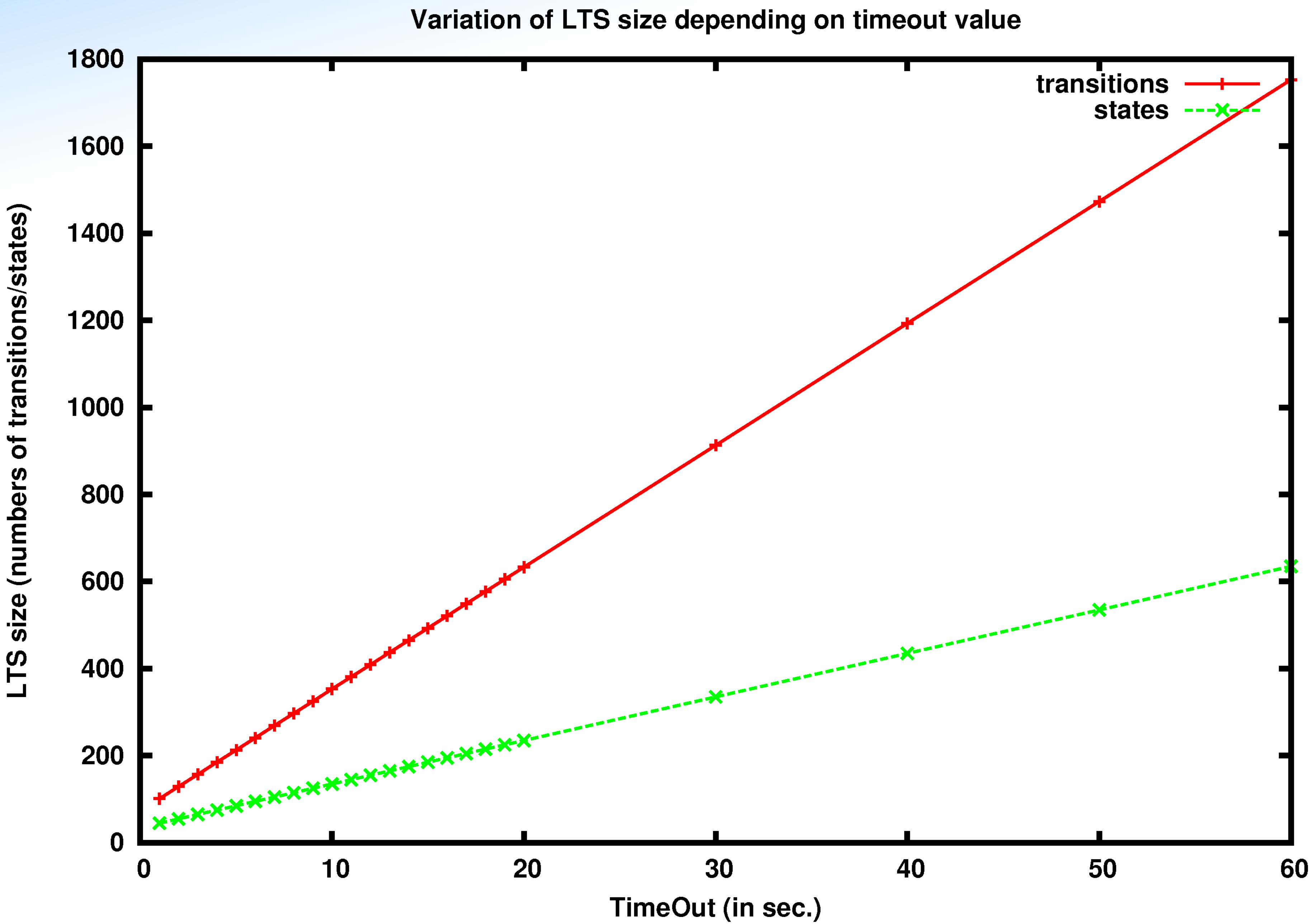
- **The main loop is guarded by a kind of “ping alive”**
  - the user can't stay in *navigation mode* or *configuration mode* more than X seconds
  - “alive” = get itinerary (for example)
- **BPEL process:**
  - Using scope activities (with maximum time execution guards and catches)

4. Discrete-time LTS analysis

LTS: initialization phase



LTS: variation of the dtLTS size



Verification of discrete-time properties

Safety Properties

- **Safety Properties**
  - specify informally that “something bad never happens”

S_1	$[\ (\neg !LoginOk)^*.?setPosition \vee ?setDestination\ ]$ false
S_2	$[\ (\mathsf{true}^*.$ $((!getPosition.(\neg ?setPosition)^*) \mid (!getDestination.(\neg ?setDestination)^*)).$ $!GoodJourney)\ ]$ false
S_3	$[\ \mathsf{true}^*.?getItinerary.(\neg(!Itinerary \vee !Destination))^*.$ $(?getPicture \vee ?getRoadMap \vee ?configMode \vee$ $?setPosition \vee ?setDestination \vee ?getItinerary)\ ]$ false
S_4	$[\ \mathsf{true}^*.?getItinerary.(\neg(!Itinerary \vee !Destination))^*.$ $(time.(\neg(!Itinerary \vee !Destination))^*)\{51\}.$ $(!Itinerary \vee !Destination)\ ]$ false

Verification of discrete-time properties

Liveness Properties

- **Liveness Properties**
  - specify informally that “something good eventually happens”

L_1	$[\ \mathsf{true}^*.!LoginOk\ ]\ \mathsf{AF}\ \langle !getPosition \vee !getDestination \vee !GoodJourney \rangle\ \mathsf{true}$
L_2	$[\ \mathsf{true}^*.!GoodJourney.(\tau \vee time)^*\ ]\ \langle (\tau \vee time)^*.\ !Ready \vee !bye \rangle\ \mathsf{true})$
L_3	$[\ \mathsf{true}^*.!Ready.\ time\{\ \ldots 50\ \}\ ]$ $\langle \mathsf{true}^*.!Picture \vee !RoadMap \vee !Itinerary \vee !Destination \rangle\ \mathsf{true}$
L_4	$[\ \mathsf{true}^*.!Ready.$ $((\neg(!Itinerary \vee !Destination \vee !Picture \vee !RoadMap))^*.time)\{51\}\ ]$ $\mathsf{AF}\ \langle !ConnectionError \rangle\ \mathsf{true}$

5. Conclusion

Conclusion

- **In domain of SOA, we propose a tool-equipped methodology for:**
  - **modeling Web services**
    - WSMod tool
    - exhaustive simulation algorithm
    - based on a formalization of BPEL semantics (process algebraic rules)
  - **analyzing Web services**
    - EVALUATOR 4.0 (from CADP toolbox)
    - Discrete-time safety and liveness properties
- **Illustrated on the example of a GPS Web service**

Future Work

- **Improve the connection between WSMod and CADP**
  - producing implicit dtLTS (according to the interface defined by Open/Caesar)
  - enable on-the-fly verification
- **Use continuous time models**
  - Lead to state explosion in presence of numerous timeouts
  - Connecting the time automata produced by WSMod with the UPPAAL tool
- **Handle compositions of multiple Web services**
  - Using tools such as Exp.Open

Formal Modeling and Discrete-Time Analysis of BPEL Web Services

EOMAS 2008

June 16-17, 2008

Sylvain Rampacek
Sylvain.Rampacek@u-bourgogne.fr
LE2I (UMR CNRS 5158) - A5

Radu Mateescu
Radu.Mateescu@inria.fr
INRIA - VASY

UNIVERSITÉ DE BOURGOGNE

Laboratoire Electronique Informatique et Image