A formal semantics for Web Services interaction

MeFoSyLoMa

June 3rd, 2005

Sylvain Rampacek
sylvain.rampacek@univ-reims.fr
CReSTIC – LAMSADE

PhD thesis, supervisors :

Serge Haddad – LAMSADE
Patrice Moreaux – CReSTIC

Summary

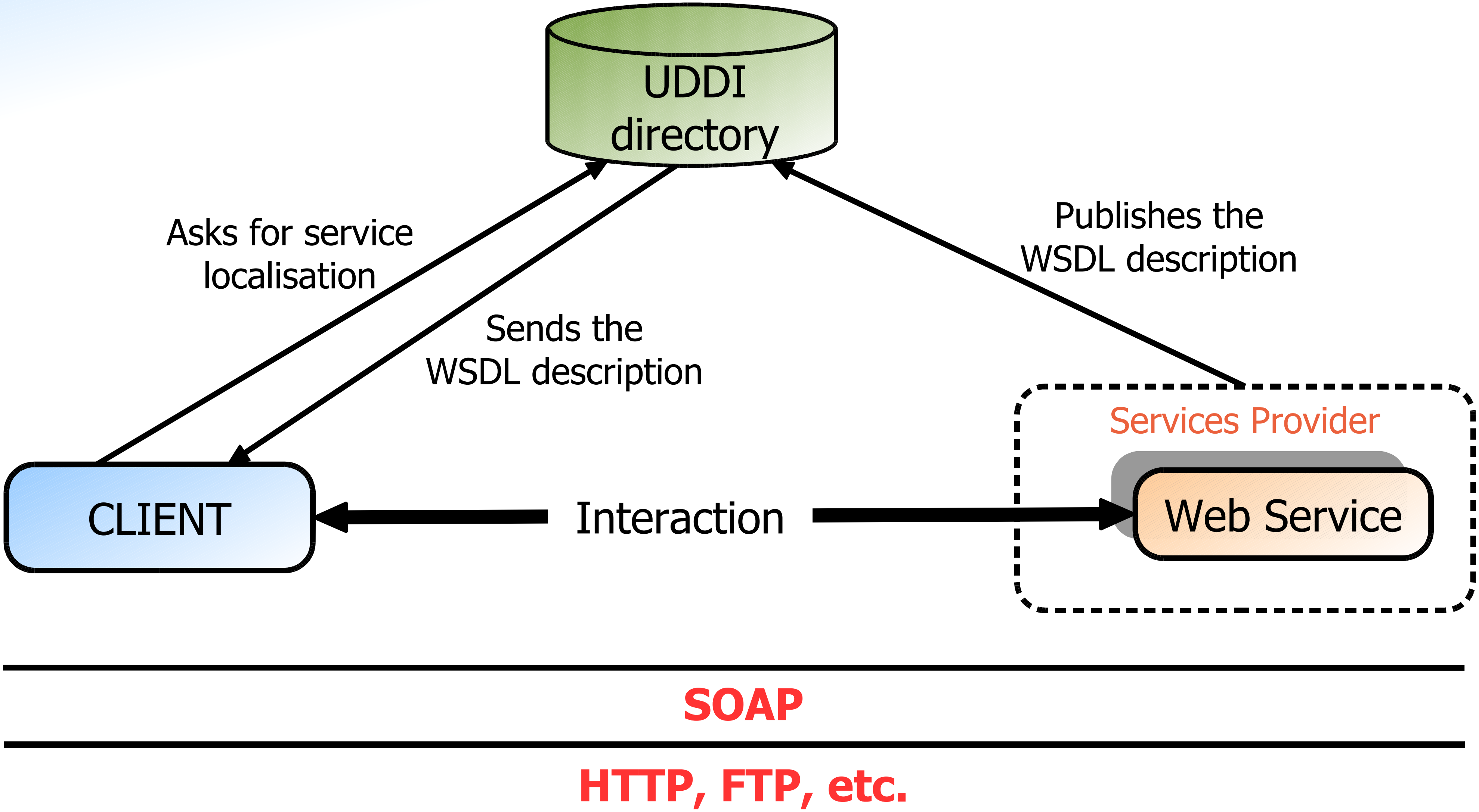
- Introduction
  - Web Services
  - Platform description
- Business Process Languages
  - XLANG and BPEL4WS
  - Formal Semantics
- Interaction Relation
  - Client Generation
  - From Discrete Time to Dense Time
- Conclusion

Introduction

Web Services Context

- Distributed system (a service ≠ a server)
- Interoperability (XML, SOAP, WSDL, ...)
- Heterogeneous management :
    - Supplementary level (keep the business level)
    - Evolution of object based distributed systems
    - Service oriented architecture (SOA)

Web Services Architecture



UDDI directory

Asks for service localisation

Sends the WSDL description

Publishes the WSDL description

CLIENT

Interaction

Services Provider

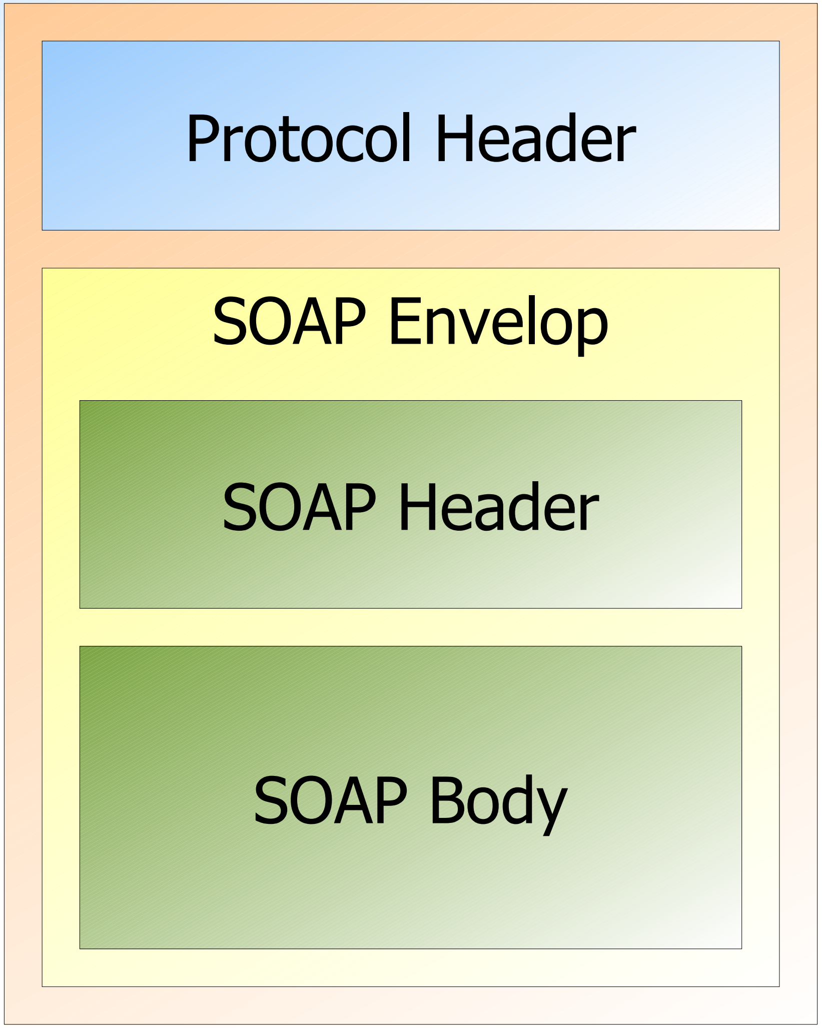
Web Service

SOAP

HTTP, FTP, etc.

SOAP

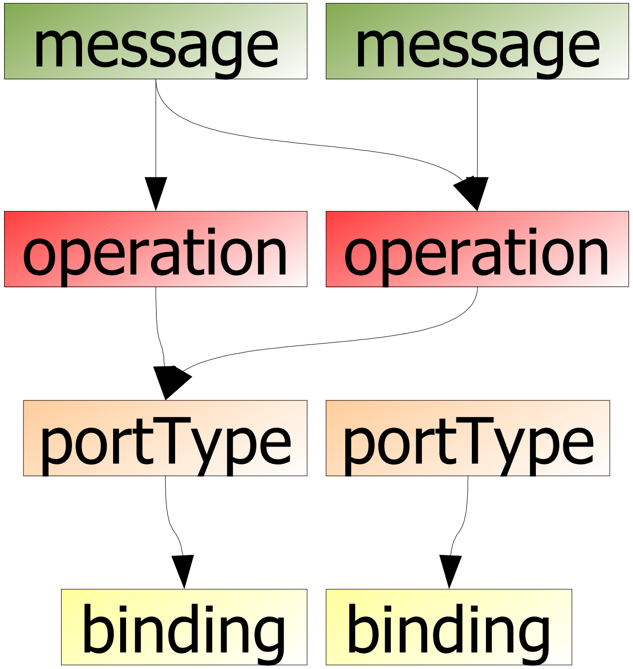
- Simple Object Access Protocol -



- Represents data
- XML Based
- 2 parts :
  - protocol header : for the transport level
  - SOAP Envelop :
    - SOAP Header : intermediary nodes and their roles
    - SOAP Body : “data” in specific language (e.g. RPC)

WSDL

- Web Services Description Language -



- A kind of interface of the service
- XML based
- Describes :
  - name-spaces
  - messages
  - operations (input and output messages composition)
  - portType (communication port)
  - binding (link WSDL-operation to SOAP-operation)
  - ... (extensibility)

Services Oriented Architectures ?

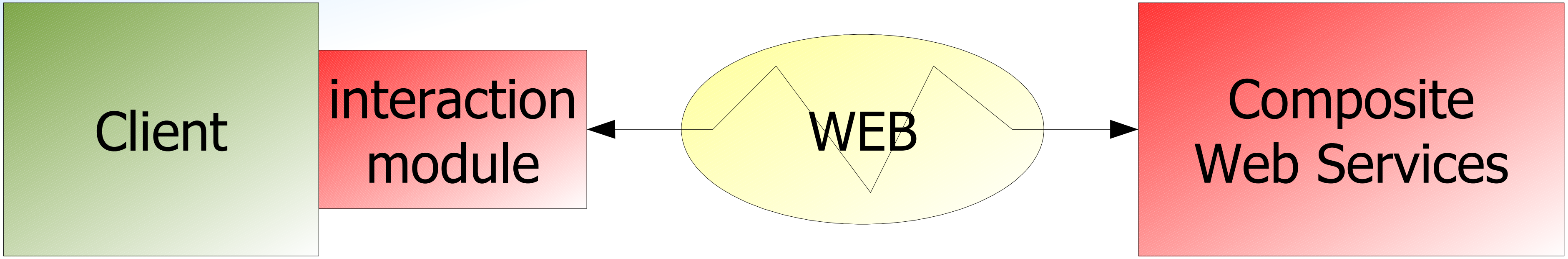
- Business Process language based on “elementary” Web Services [MonfortGoudeau2004]
- Extension of WSDL description
  - XLANG, BPML, BPEL4WS...
- Composition of Web Services
- Problem : semantics, especially coordination (orchestration or choreography)

Interaction relation between clients and services

Orchestration or Choreography ?

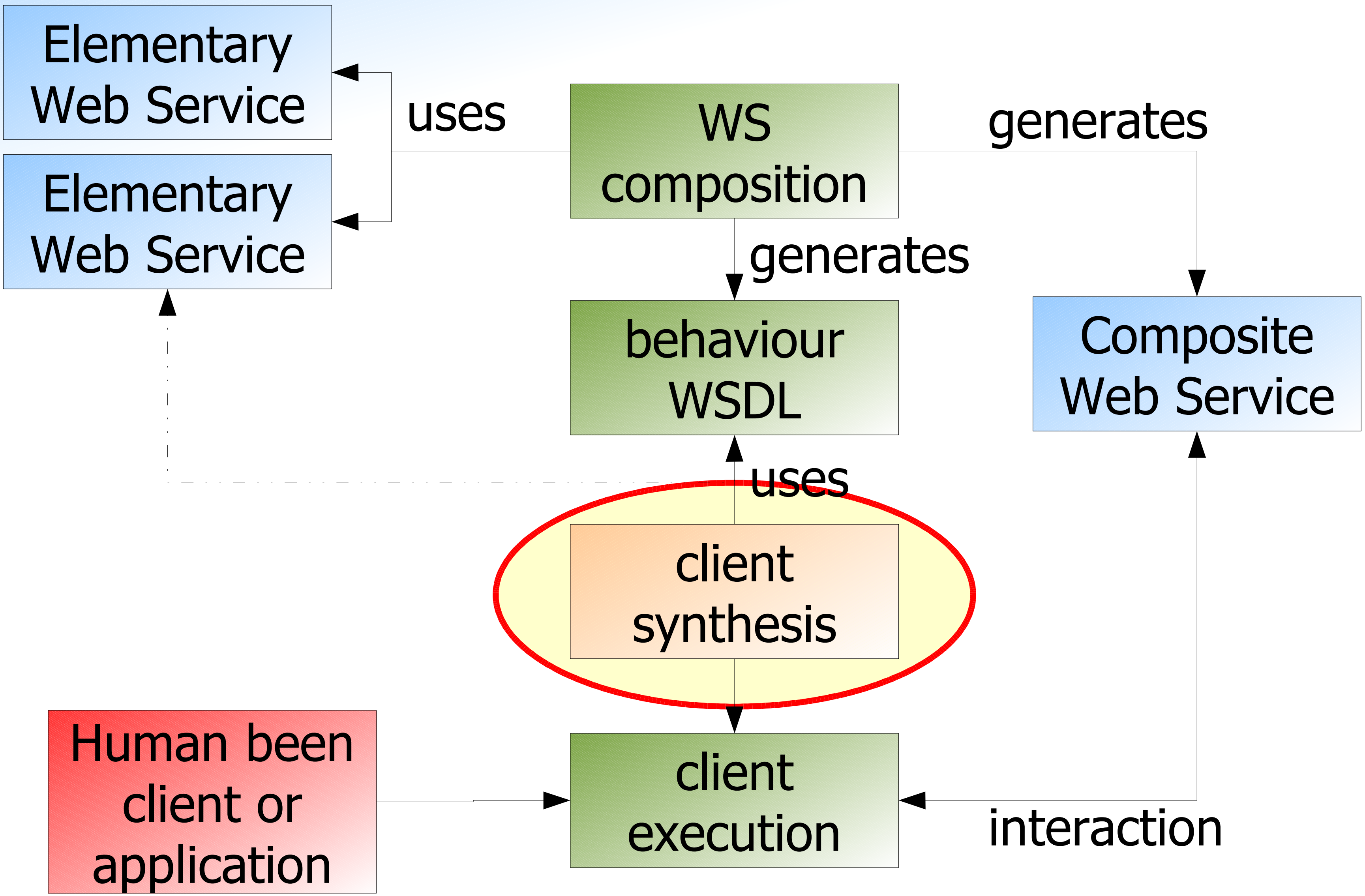
- Orchestration :
  - central process :
    - takes control and coordinates operations of the involved Web Services
    - Web Services do not know that they are involved into a composition
- Choreography :
  - does not rely on a central coordinator :
    - Web Service knows exactly when to execute its operations and whom interact with
    - collaborative effort focused on exchange of messages

Context of our work



- Development platform for composite Web Services
  - Orchestration method

Composite Web Services Platform



Business Process Language
and
Formal Semantics

Business Process Languages versus WSDL

- WSDL
  - describes the interface of Web Services
  - does not describe the behaviour of the service
- Business Process Description Languages
  - describe interaction flows
  - describe semantics and/or behaviour of the business processes

Business Process Languages

- XLANG (Microsoft) :
  - Basic elements : action, while, switch, context, ...
    - Conditions
    - Loops
    - Time and exceptions managements
- BPEL4WS (IBM, BEA, Microsoft) :
  - Merge of XLANG and BPML
    - advantages of both (basic elements, flow management and more)
  - Cancelling mechanism (compensate)
    - useful for long interaction (several days)

Formal Semantics

- Required for adapted client construction
  - Formal composition
  - Interaction relation (timed automaton)
  - Controlled client generation
- 2 aspects :
  - Algebra of Timed Processes (ATP)
  - Associated Semantics
    - TIOTS (discrete time)
    - Timed Automaton (dense time)

XLANG and ATP

- XLANG formalization with ATP [HMMR04a]
  - remove ambiguity to XLANG language
  - use a generic method
- Actions of the processes :

Send/Receive message	$!o[m]$ / $?o[m]$
Time passing (discrete time)	χ
Internal action	τ
Terminate action	$\surd$

XLANG/ATP – *formal semantics*

basic processes – *examples*

time process

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

empty process

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

operation process

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

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

XLANG/ATP – *formal semantics*

advanced processes – *examples*

sequence process

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

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

while process

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

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

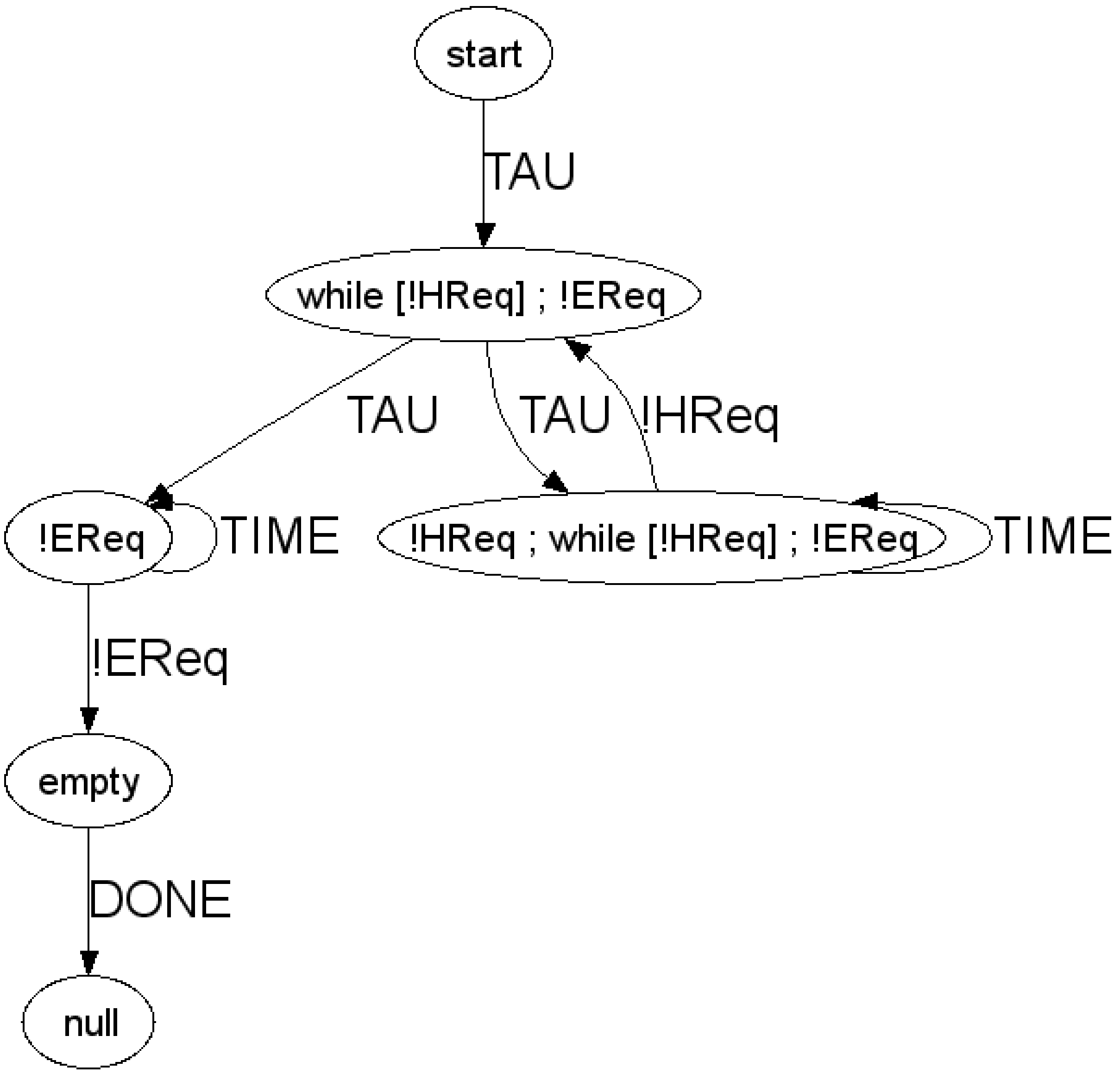
BPEL4WS migration

- Business Process Execution Language for Web Services -

- Generic method usefulness
- Same basic elements (*some names change*)
- Element *actions/operations* are now (link to WSDL operations) :
  - *receive* (and *reply* if necessary)
  - *invoke*
- New functionality (will be implemented later) :
  - process *flow* : *links* mechanism
  - all process : *compensate*

Example
TIOTS – Service Side

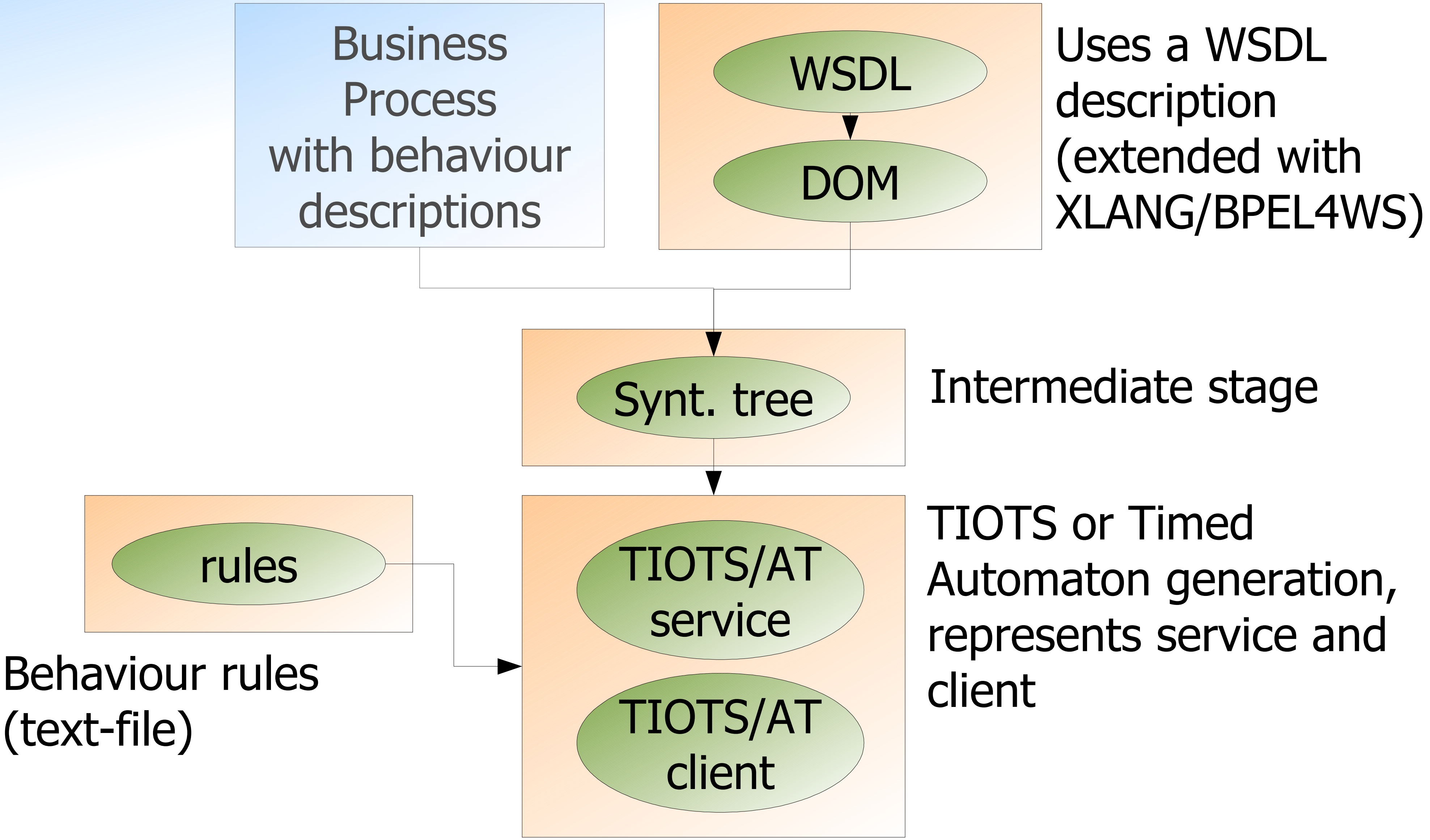
Process :
while {!HReq} ; !EReq



Tools
Generic TIOTS synthesis

- Need for “generic” synthesis :
  - Don't be linked to only one language
  - Adaptability of the behaviour and the semantics
- A generator using rules files :
  - Each basic elements is described by :
    - Guards
    - Results transitions (and target state)
    - Rewriting rules (merging states)

Organization of the Synthesis



Business Process with behaviour descriptions

WSDL

DOM

Uses a WSDL description (extended with XLANG/BPEL4WS)

Synt. tree

Intermediate stage

rules

Behaviour rules (text-file)

TIOTS/AT service

TIOTS/AT client

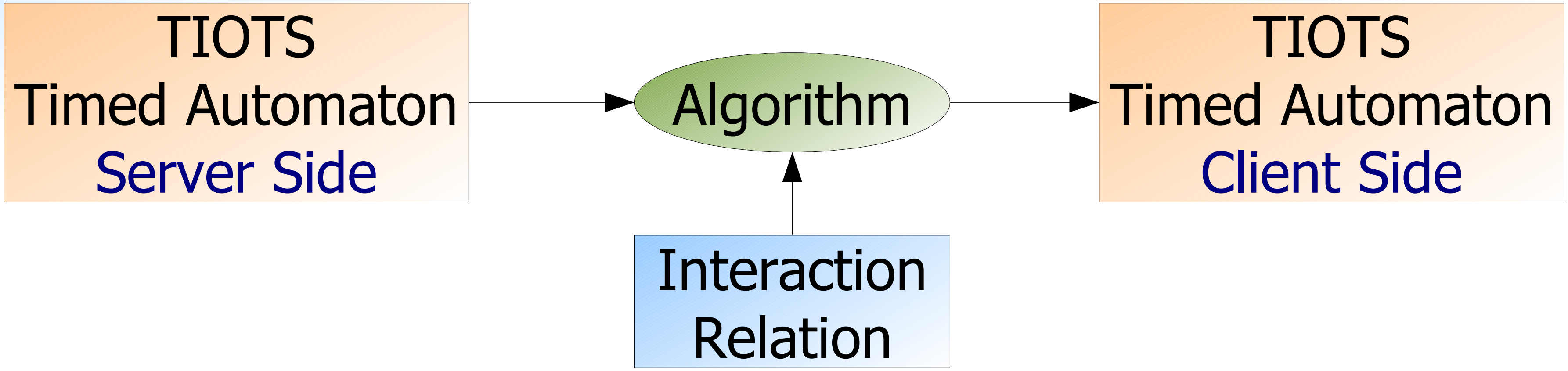
TIOTS or Timed Automaton generation, represents service and client

Interaction Relation

Interaction Relation

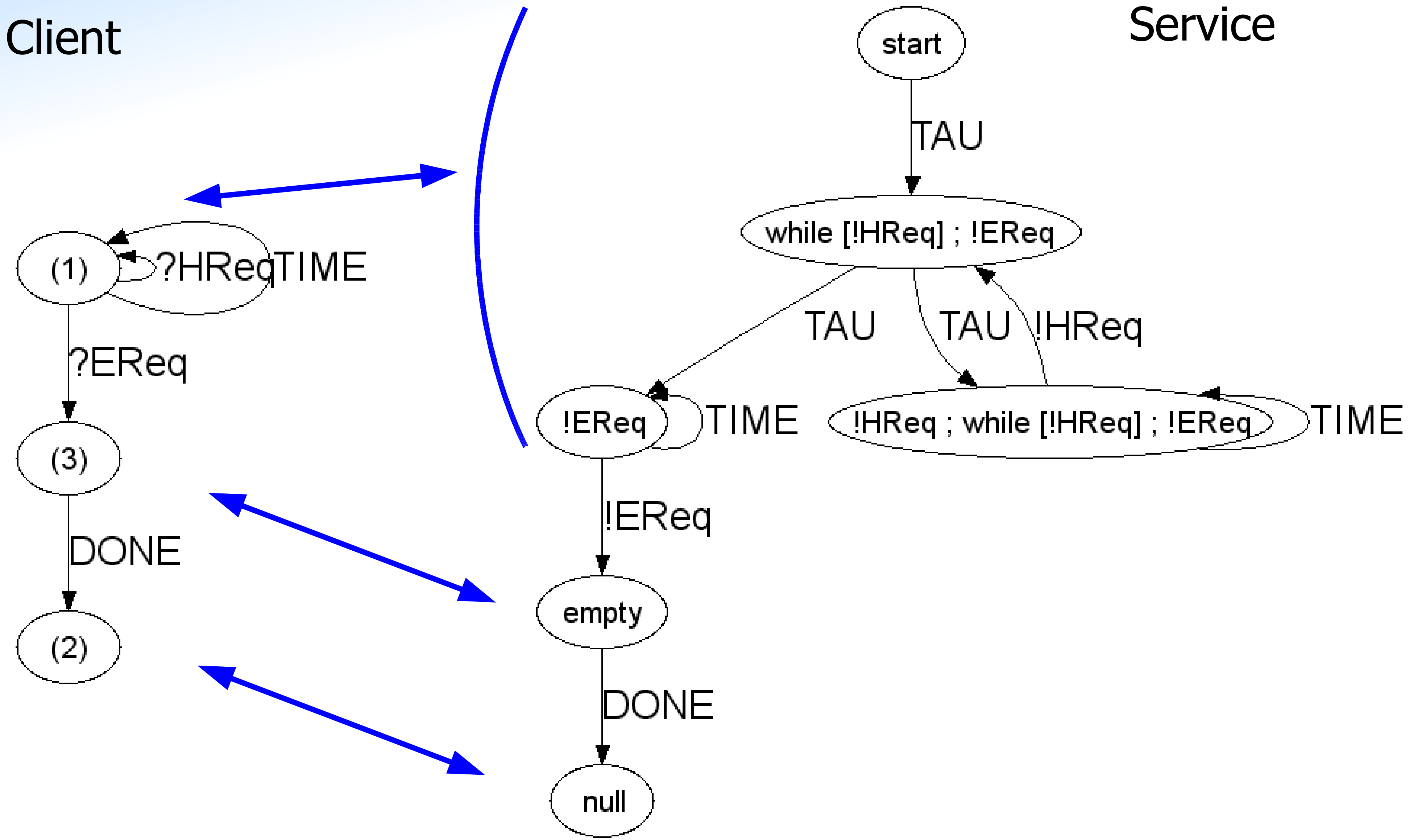
- Once the service side TIOTS is generated, we use an interaction relation to generate the client side
- Adapted *Interaction relation* between a client and a service :
  - If a message is sent by a service, then the client must be able to receive a message
  - If a service is waiting a message, then the client must be able to send it
  - If the server sees time passing, then the client must also see time passing

Client Generation



- Algorithm :
  - based on determinization-like of the server's TIOTS
  - a client state is the “*internal-action*”-closure of service state
  - ambiguity detection

Example
TIOTS – Client side



From Discrete Time to Dense Time

From discrete time to dense time

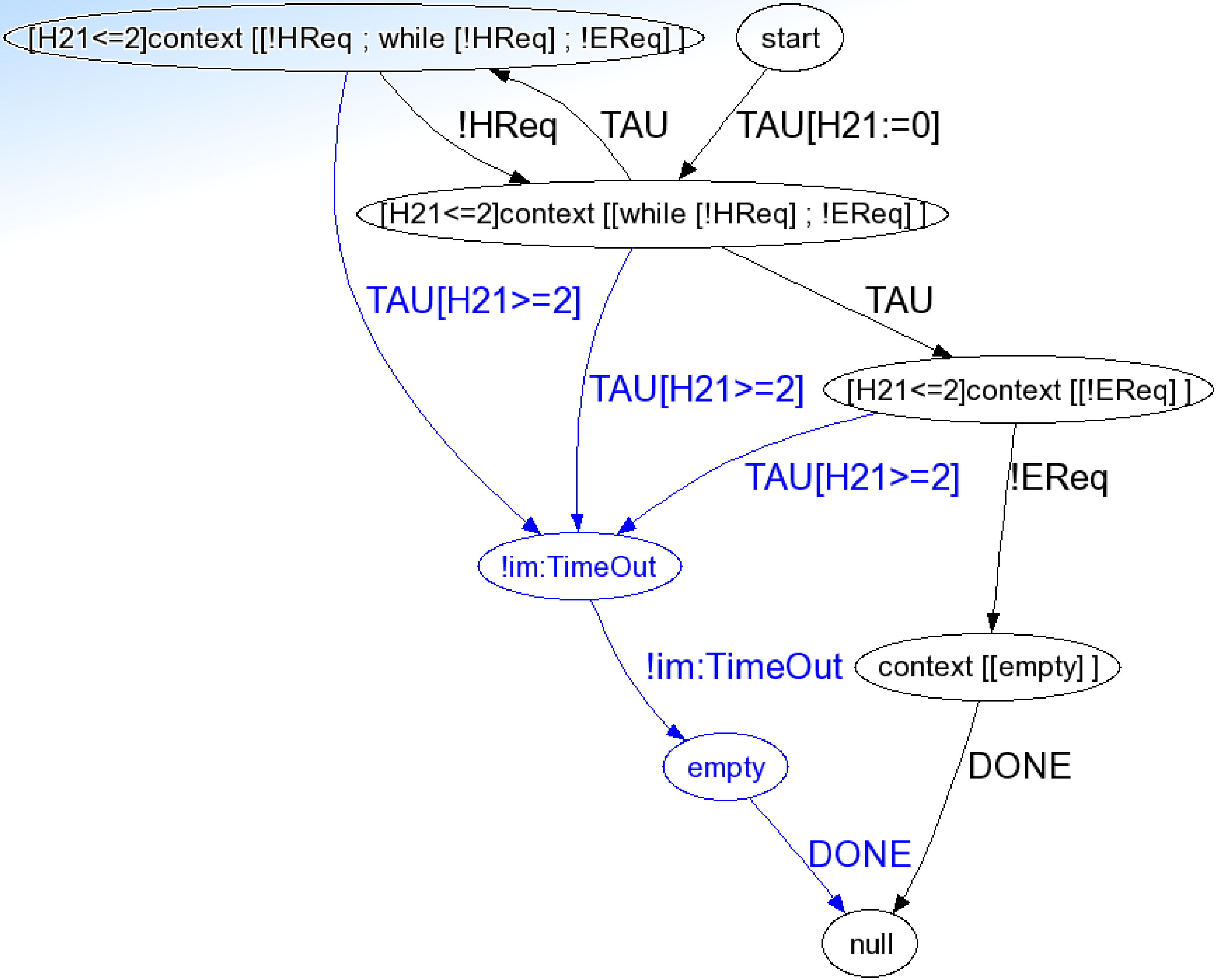
- Time passing is reflected by one transition in the TIOTS.
- In case of complex Web Services (with imbrication of different maximal execution times), the number of states explodes !

=> Switch to dense time semantics

Second model
– dense time –

- From TIOTS to TA [HMMR04b]
  - Semantics adaptation :
    - Delete explicit time passing
    - Add guards to transition
    - Add invariants to state
  - Problem : un-decidability for determinization of general timed automaton

Example – Timed Automaton – Service Side



Conclusion

Conclusion

- Client-Service Interaction relation (in discrete time)
- Generic client synthesis tools for Web Services (elementary or composite)
- Emerging concepts and moving technologies

  => generic tools

Perspectives

- Dense time interaction relation and client's TA algorithm generation
  - Uses adapted classes of timed automaton
- Under development :
  - client's interaction module : invocation of Web Service based on client's TIOTS / TA
  - server side composition tools for Web Services (orchestration)
- Final goal : a platform to “validate” and orchestrate service oriented applications.

Bibliography

- [HMMR04a] S. Haddad, T. Melliti, P. Moreaux, and S. Rampacek. *A dense time semantics for Web services specifications languages* (ICTTA'04).
- [HMMR04b] S. Haddad, T. Melliti, P. Moreaux, and S. Rampacek. *Modelling web services interoperability* (ICEIS04).
- [MonfortGoudeau2004] V. Monfort, S. Goudeau. *Web services et interopérabilité des SI*. Ed. Dunod 2004.