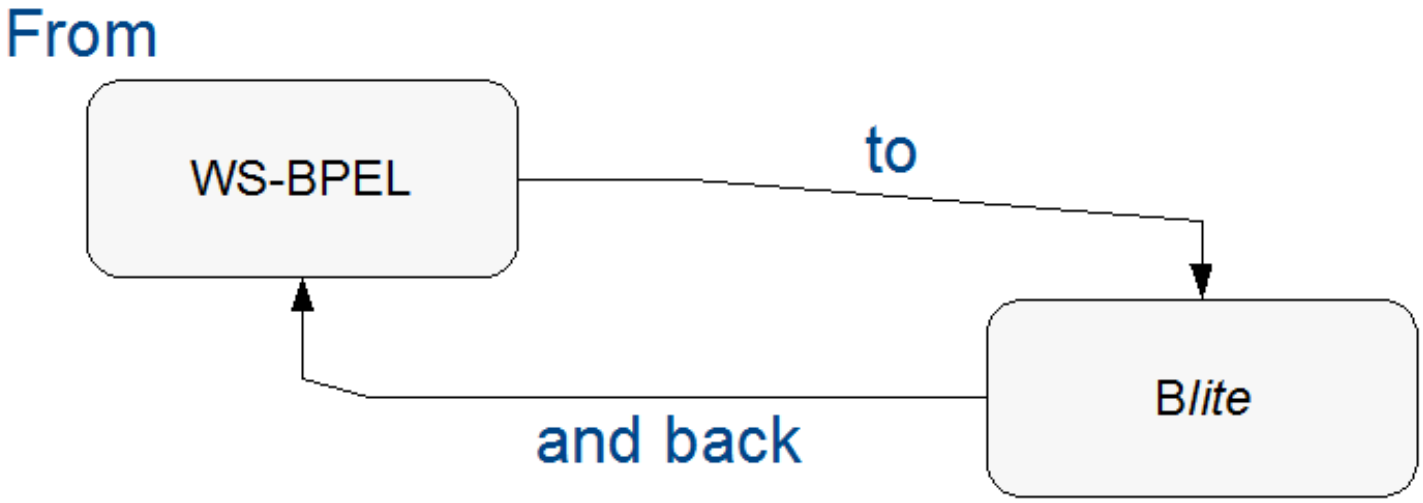


Francesco Tiezzi

Joint work with
L. Cesari, A. Lapadula, P. Panconi, R. Pugliese
Dipartimento di Sistemi e Informatica
Università degli Studi di Firenze

Sensoria Meeting, Lisbon — June 9, 2009

Outline

1. A glimpse of WS-BPEL
2. From WS-BPEL to B*lite* ...
3. ...and back (tools description and demo)
4. Concluding remarks

WS-BPEL

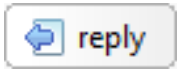
- Web Services Business Process Execution Language Version 2.0
- Is an OASIS standard (11 April 2007)
- Is the standard language for orchestration of Web Services
- Relies on the following XML-based specifications
  - WSDL for interface
  - XML Schema for types
  - XPath for expressions

WS-BPEL: basic activities

- 

to do nothing
- 

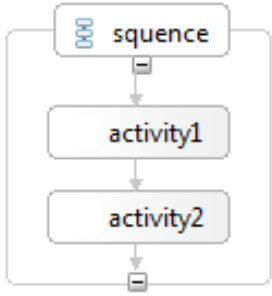
to invoke an operation offered by a (partner) service
  - partner services are identified by *partner links*, defining the shape of peer-to-peer conversational relationships
- 

to wait for a matching message to arrive
- 

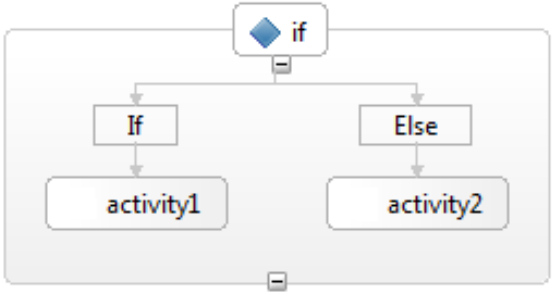
to send a message in reply to a message that was received by a receive activity
- 

to update the values of variables with new data

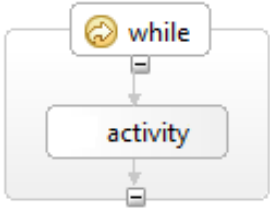
WS-BPEL: control flow activities



to perform a collection of activities
in sequential order



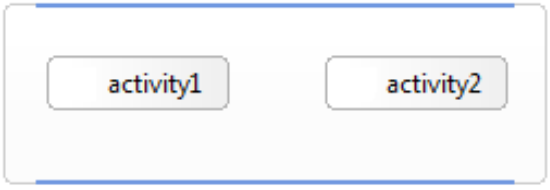
to select exactly one activity for execution
from a set of choices



to repeat an activity as long as
a specified condition is true

WS-BPEL: control flow activities

to wait for one of several possible messages to arrive

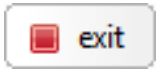


to concurrently perform a collection of activities

WS-BPEL: fault and compensation

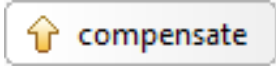
- *Fault handling*: similar to exception handling of 'classic' programming languages
- *Compensation*: execution of specific activities (attempting) to reverse the effects of previously executed activities

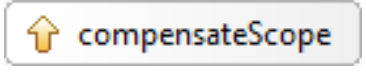
WS-BPEL: fault and compensation

- *Fault handling*: similar to exception handling of 'classic' programming languages
- *Compensation*: execution of specific activities (attempting) to reverse the effects of previously executed activities
- 

to immediately end a business process instance
- 

to generate a fault from inside an instance
- 

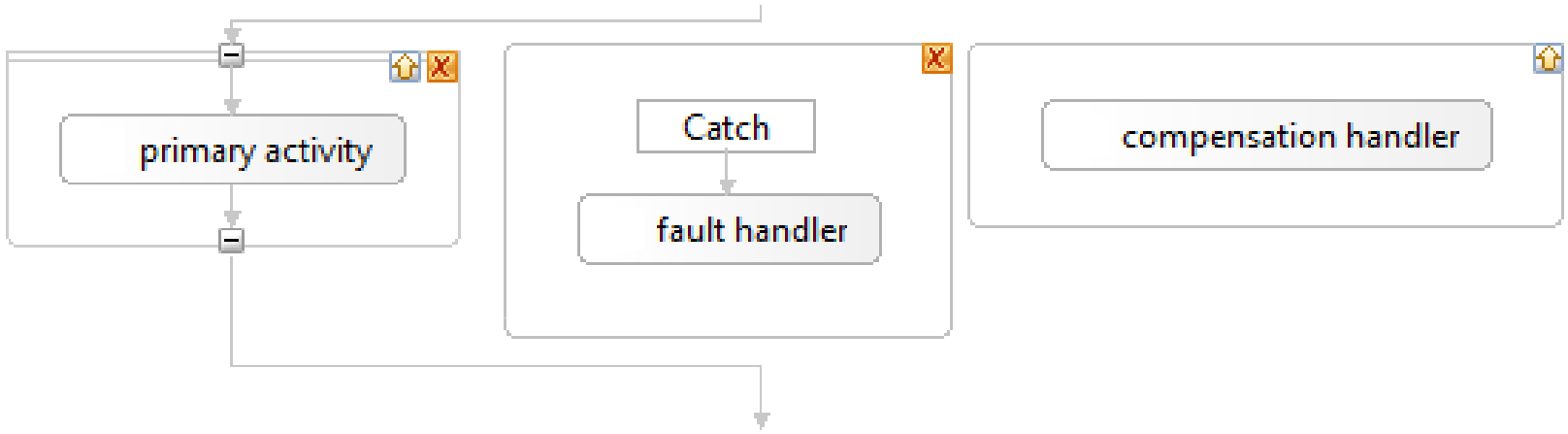
to rethrow the fault that was originally caught by the immediately enclosing fault handler
- 

to start compensation on all inner scopes that have already completed successfully, in the *reverse order* of completion
- 

to start compensation on a specified inner scope that has already completed successfully

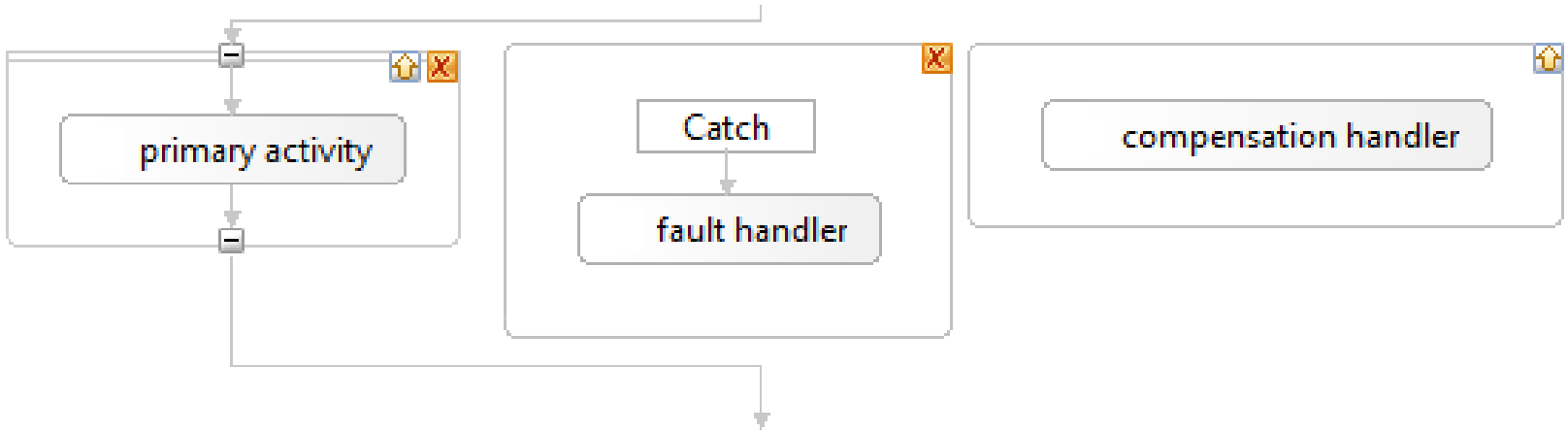
WS-BPEL: fault and compensation

- *Scope activity*: groups a primary activity together with fault handling activities and a compensation handling activity



WS-BPEL: fault and compensation

- *Scope activity*: groups a primary activity together with fault handling activities and a compensation handling activity



- We left out the following aspects of WS-BPEL
  - termination and event handlers within scope activities
  - synchronization dependencies within flow activities
  - repeatUntil and forEach activities
  - wait activities

WS-BPEL engines

- Three of the most known freely available WS-BPEL engines

Oracle BPEL Process Manager 10.1.3

`http://www.oracle.com/technology/bpel`

ActiveBPEL 4.1

`http://www.activevos.com`

Apache ODE 1.1.1

`http://ode.apache.org`

Outline

1. A glimpse of WS-BPEL
2. From WS-BPEL to B*lite* ...
3. ...and back (tools description and demo)
4. Concluding remarks

From WS-BPEL to B*lite*

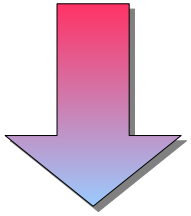
Developing WS-BPEL applications is difficult and error-prone

- WS-BPEL has an XML syntax
- WS-BPEL is equipped with intricate features
- WS-BPEL comes without a formal semantics
  - its specification document is written in natural language
  - engines implement different semantics [COORDINATION08]

From WS-BPEL to B*lite*

Developing WS-BPEL applications is difficult and error-prone

- WS-BPEL has an XML syntax
- WS-BPEL is equipped with intricate features
- WS-BPEL comes without a formal semantics
  - its specification document is written in natural language
  - engines implement different semantics [COORDINATION08]



B*lite* (BPEL lite)

A lightweight (formal) language designed around some of WS-BPEL distinctive features

B*lite*: a lightweight variant of WS-BPEL

- Is designed around some of WS-BPEL distinctive features
  - E.g. partner links, process termination, message correlation, long-running transactions and compensation
  - To keep the design of the language manageable, other WS-BPEL aspects have been left out:
    timeouts, event, termination handlers, flow graphs, data handling
- Permits expressing WS-BPEL business processes, instances and deployments
- Is equipped with a formal operational semantics defined in terms of a structural congruence and a reduction relation
  - the semantics describes the behaviour of an *abstract engine* for WS-BPEL processes

B*lite*: syntax

- Basic activities

  $b ::= \mathrm{inv}\,\ell^{i}\,o\,\bar{x} \mid \mathrm{rcv}\,\ell^{r}\,o\,\bar{x} \mid x := e \mid \mathrm{empty} \mid \mathrm{throw} \mid \mathrm{exit}$

(Notations)

ℓ^{i}	invoke partner links
ℓ^{r}	receive partner links
o	operations
x	variables
$\bar{x}$	tuples of variables
e	expressions

B*lite*: syntax

- Basic activities

$$b ::= inv\,\ell^{i}\,o\,\bar{x} \mid rcv\,\ell^{r}\,o\,\bar{x} \mid x := e \mid empty \mid throw \mid exit$$

(Notations)

ℓ^{i}	invoke partner links
ℓ^{r}	receive partner links
o	operations
x	variables
$\bar{x}$	tuples of variables
e	expressions

Information Society Technologies

B*lite*: syntax

- Basic activities

  $$\mathsf{b} ::= \mathsf{inv}\,\ell^{i}\,\mathsf{o}\,\bar{\mathsf{x}} \mid \mathsf{rcv}\,\ell^{r}\,\mathsf{o}\,\bar{\mathsf{x}} \mid \mathsf{x} := \mathsf{e} \mid \mathsf{empty} \mid \mathsf{throw} \mid \mathsf{exit}$$

- Structured activities

  $$\begin{array}{rcl} \mathsf{a} & ::= & \mathsf{b} \mid \mathsf{if(e)\{a_1\}\{a_2\}} \mid \mathsf{while(e)\,\{a\}} \mid \mathsf{a_1\,;\,a_2} \\ & \mid & \sum_{j\in J} \mathsf{rcv}\,\ell_j^{r}\,\mathsf{o}_j\,\bar{\mathsf{x}}_j\,;\,\mathsf{a}_j \mid \mathsf{a_1} \,|\, \mathsf{a_2} \mid [\mathsf{a} \bullet \mathsf{a}_f \star \mathsf{a}_c] \end{array}$$

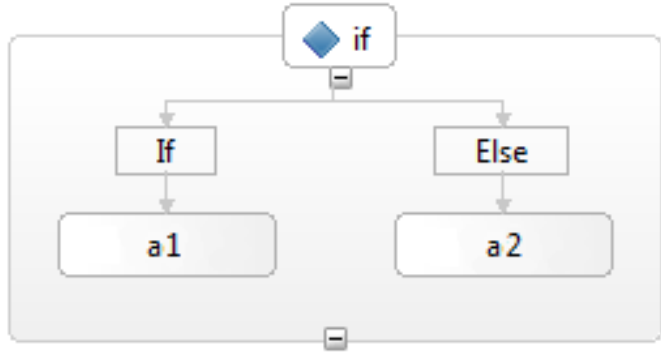
B*lite*: syntax

- Basic activities

  $b ::= \mathsf{inv}\,\ell^{i}\,o\,\bar{x} \mid \mathsf{rcv}\,\ell^{r}\,o\,\bar{x} \mid x := e \mid \mathsf{empty} \mid \mathsf{throw} \mid \mathsf{exit}$

- Structured activities

  $a ::= b \mid \mathsf{if}(e)\{a_1\}\{a_2\} \mid \mathsf{while}(e)\{a\} \mid a_1 ; a_2$
  $\mid \sum_{j\in J} \mathsf{rcv}\,\ell_j^{r}\,o_j\,\bar{x}_j ; a_j \mid a_1 \mid a_2 \mid [a \bullet a_f \star a_c]$



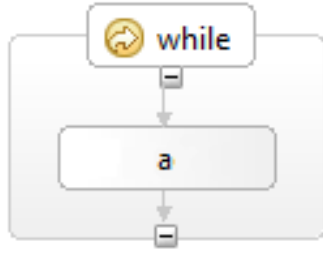
B*lite*: syntax

- Basic activities

  $$b ::= \mathsf{inv}\,\ell^{i}\,o\,\bar{x} \mid \mathsf{rcv}\,\ell^{r}\,o\,\bar{x} \mid x := e \mid \mathsf{empty} \mid \mathsf{throw} \mid \mathsf{exit}$$

- Structured activities

  $$a ::= b \mid \mathsf{if}(e)\{a_1\}\{a_2\} \mid \mathsf{while}(e)\,\{a\} \mid a_1\,;\,a_2$$
  $$\mid \sum_{j\in J} \mathsf{rcv}\,\ell_j^{r}\,o_j\,\bar{x}_j\,;\,a_j \mid a_1 \mid a_2 \mid [a \bullet a_f \star a_c]$$



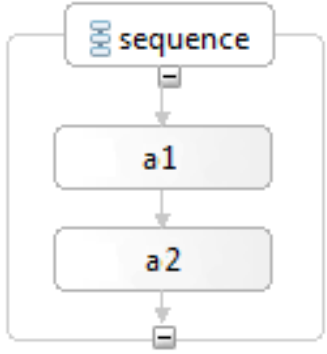
B*lite*: syntax

- Basic activities

  $b ::= \mathsf{inv}\,\ell^{i}\,o\,\bar{x} \mid \mathsf{rcv}\,\ell^{r}\,o\,\bar{x} \mid x := e \mid \mathsf{empty} \mid \mathsf{throw} \mid \mathsf{exit}$

- Structured activities

  $a ::= b \mid \mathsf{if}(e)\{a_1\}\{a_2\} \mid \mathsf{while}(e)\{a\} \mid a_1 ; a_2$
  $\mid \sum_{j\in J} \mathsf{rcv}\,\ell_j^{r}\,o_j\,\bar{x}_j ; a_j \mid a_1 \mid a_2 \mid [a \bullet a_f \star a_c]$



B*lite*: syntax

- Basic activities

  $b ::= \mathsf{inv}\,\ell^{i}\,o\,\bar{x} \mid \mathsf{rcv}\,\ell^{r}\,o\,\bar{x} \mid x := e \mid \mathsf{empty} \mid \mathsf{throw} \mid \mathsf{exit}$

- Structured activities

  $a ::= b \mid \mathsf{if}(e)\{a_1\}\{a_2\} \mid \mathsf{while}(e)\{a\} \mid a_1\,;\,a_2$
  $\mid \sum_{j\in J} \mathsf{rcv}\,\ell_j^{r}\,o_j\,\bar{x}_j\,;\,a_j \mid a_1 \mid a_2 \mid [a \bullet a_f \star a_c]$

B*lite*: syntax

- Basic activities

  $$\mathsf{b} ::= \mathsf{inv}\,\ell^{i}\,\mathsf{o}\,\bar{\mathsf{x}} \mid \mathsf{rcv}\,\ell^{r}\,\mathsf{o}\,\bar{\mathsf{x}} \mid \mathsf{x} := \mathsf{e} \mid \mathsf{empty} \mid \mathsf{throw} \mid \mathsf{exit}$$

- Structured activities

  $$\begin{aligned}\mathsf{a} ::= {} & \mathsf{b} \mid \mathsf{if(e)\{a_1\}\{a_2\}} \mid \mathsf{while(e)\{a\}} \mid \mathsf{a_1\,;\,a_2} \\ & \mid \textstyle\sum_{j\in J}\mathsf{rcv}\,\ell_j^{r}\,\mathsf{o}_j\,\bar{\mathsf{x}}_j\,;\,\mathsf{a}_j \mid \mathsf{a_1} \mid \mathsf{a_2} \mid [\mathsf{a} \bullet \mathsf{a}_f \star \mathsf{a}_c]\end{aligned}$$

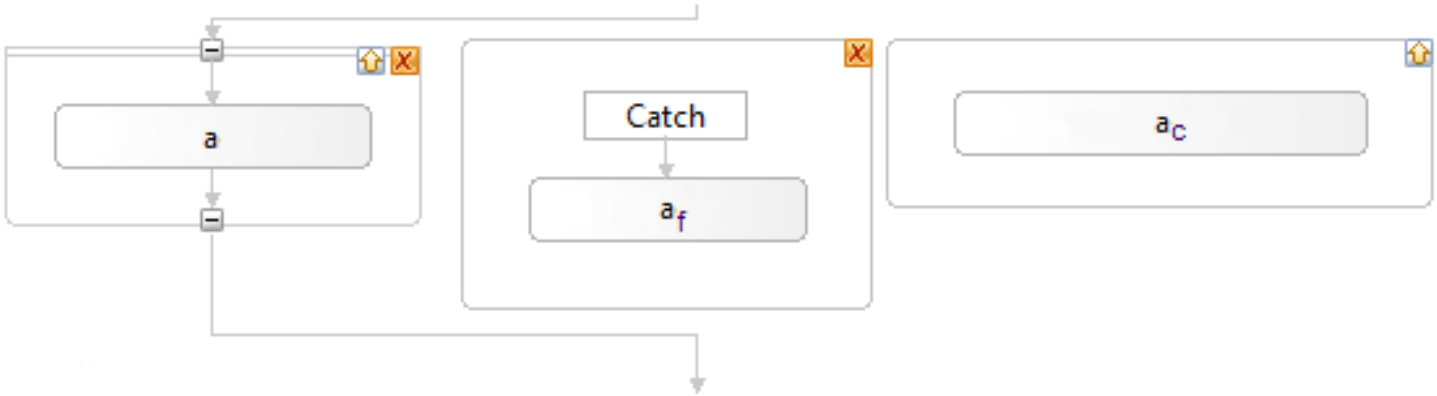
B*lite*: syntax

- Basic activities

  $b ::= \mathsf{inv}\,\ell^{i}\,o\,\bar{x} \mid \mathsf{rcv}\,\ell^{r}\,o\,\bar{x} \mid x := e \mid \mathsf{empty} \mid \mathsf{throw} \mid \mathsf{exit}$

- Structured activities

  $a ::= b \mid \mathsf{if}(e)\{a_1\}\{a_2\} \mid \mathsf{while}(e)\{a\} \mid a_1 ; a_2$
  $\mid \sum_{j\in J} \mathsf{rcv}\,\ell_j^{r}\,o_j\,\bar{x}_j ; a_j \mid a_1 \mid a_2 \mid [a \bullet a_f \star a_c]$



B*lite*: syntax

- Basic activities

  $b ::= \mathsf{inv}\,\ell^{i}\,o\,\bar{x} \mid \mathsf{rcv}\,\ell^{r}\,o\,\bar{x} \mid x := e \mid \mathsf{empty} \mid \mathsf{throw} \mid \mathsf{exit}$

- Structured activities

  $a ::= b \mid \mathsf{if}(e)\{a_1\}\{a_2\} \mid \mathsf{while}(e)\{a\} \mid a_1\,;\,a_2$
  $\mid \sum_{j\in J} \mathsf{rcv}\,\ell_j^{r}\,o_j\,\bar{x}_j\,;\,a_j \mid a_1 \,|\, a_2 \mid [a \bullet a_f \star a_c]$

- Business process definitions and instances

  $s ::= [r \bullet a_f] \mid \mu \vdash a \mid \mu \vdash a\,,\,s$

- Nets of deployed processes

  $d ::= \{s\}_c \mid d_1 \| d_2$

(Notations)

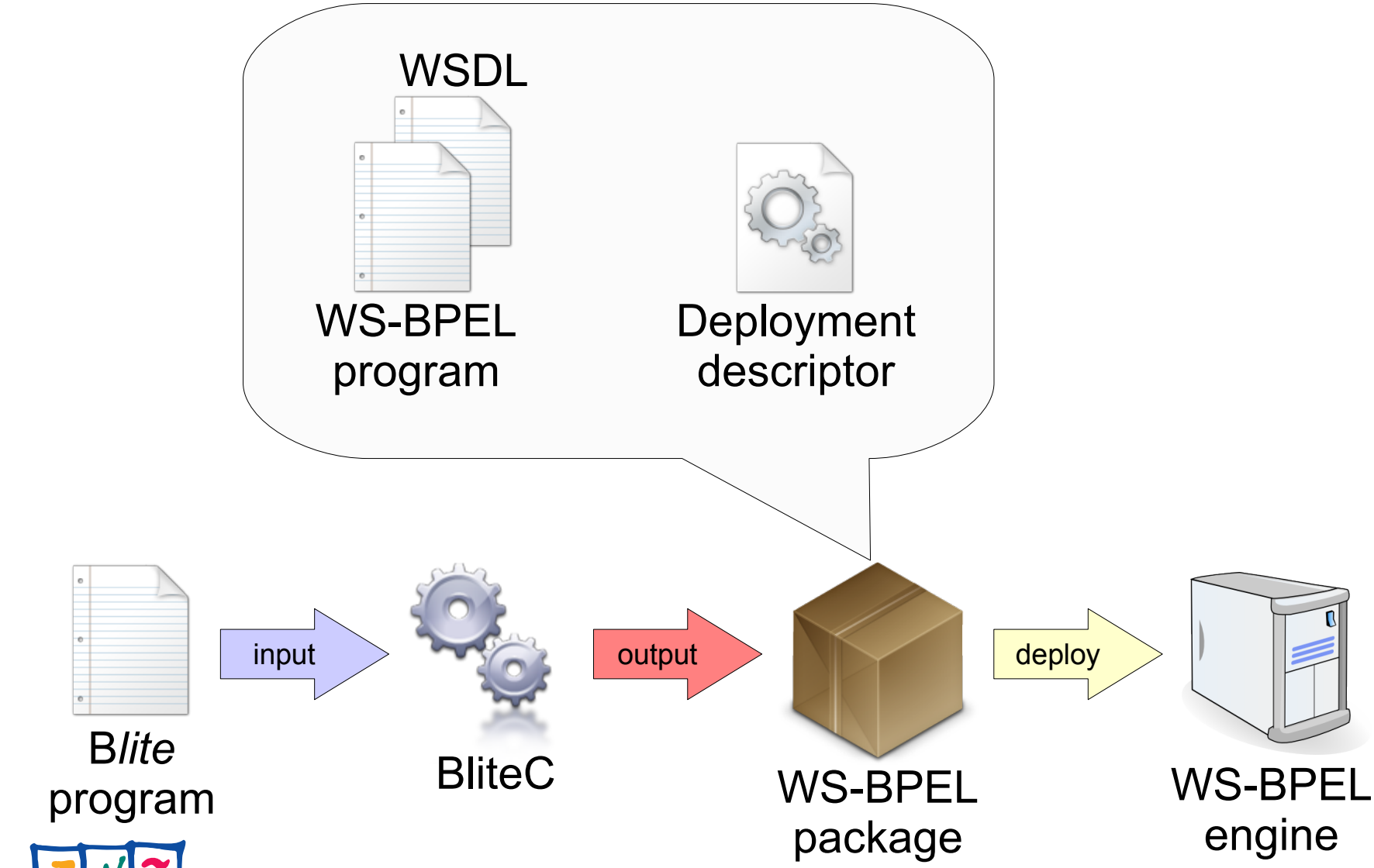
r	start activities
μ	instance states
c	correlation sets

Information Society Technologies

Outline

1. A glimpse of WS-BPEL
2. From WS-BPEL to B*lite* ...
3. ...and back (tools description and demo)
4. Concluding remarks

BliteC: from B*lite* to WS-BPEL

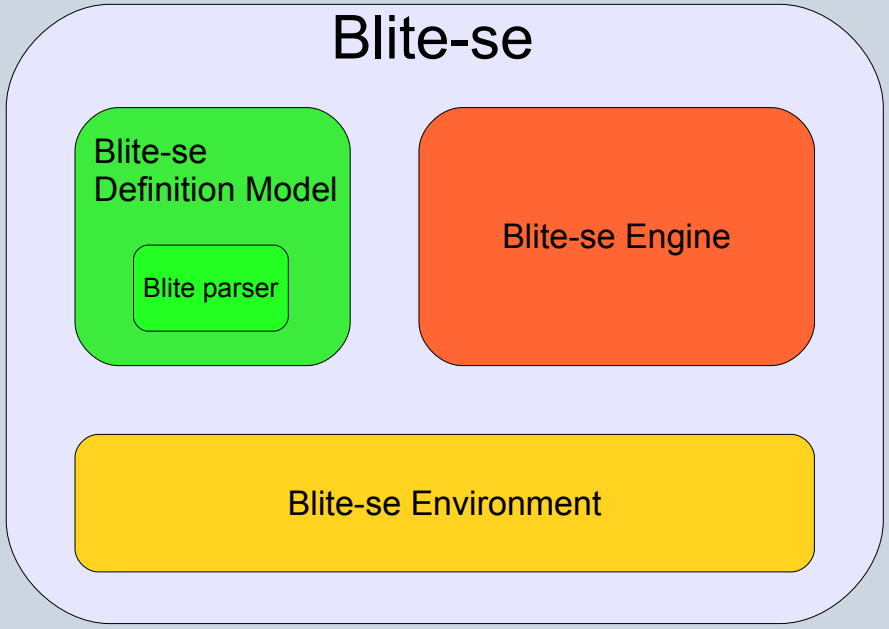


Information Society Technologies

Blite-se and Blide

Blite-se (B*lite* service engine)

- **Definition model**
  manages static definitions of B*lite* programs
- **Engine**
  executes B*lite* programs
- **Environment**
  allows the engine to abstract from communication and deployment



Blide (B*lite* integrated development environment)

permits writing and testing B*lite* programs

Tools demonstration . . .

Outline

1. A glimpse of WS-BPEL
2. From WS-BPEL to B*lite* ...
3. ...and back (tools description and demo)
4. Concluding remarks

Concluding remarks

- Undefined/ambiguous aspects of the WS-BPEL specification have led to engines implementing different semantics
  - portability of WS-BPEL over different platforms is undermined
- B*lite* formal semantics helps clarifying such aspects
- *On-going and future work*
  - **BliteC**
    - support for other WS-BPEL engines
    - integration with XPath
  - **Blite-se** and **Blide**
    - development of other Blite-se Environments
    - enhancement of Blide graphical formalism

Thank you!

For further details visit http://rap.dsi.unifi.it/blite