A reuse-based approach to the correct and automatic web service composition

Paola Inverardi and Massimo Tivoli

University of L'Aquila
Dep. Computer Science

{inverard, tivoli}@di.univaq.it

Application domain

- Distributed business processes that cross organizational boundaries
  - e-Government, e-Commerce, e-Banking
- They can be implemented by performing *service composition*
  - i.e., as novel services that *correctly orchestrate* existing services
- Handcrafted service composition is supported but it is still a difficult activity

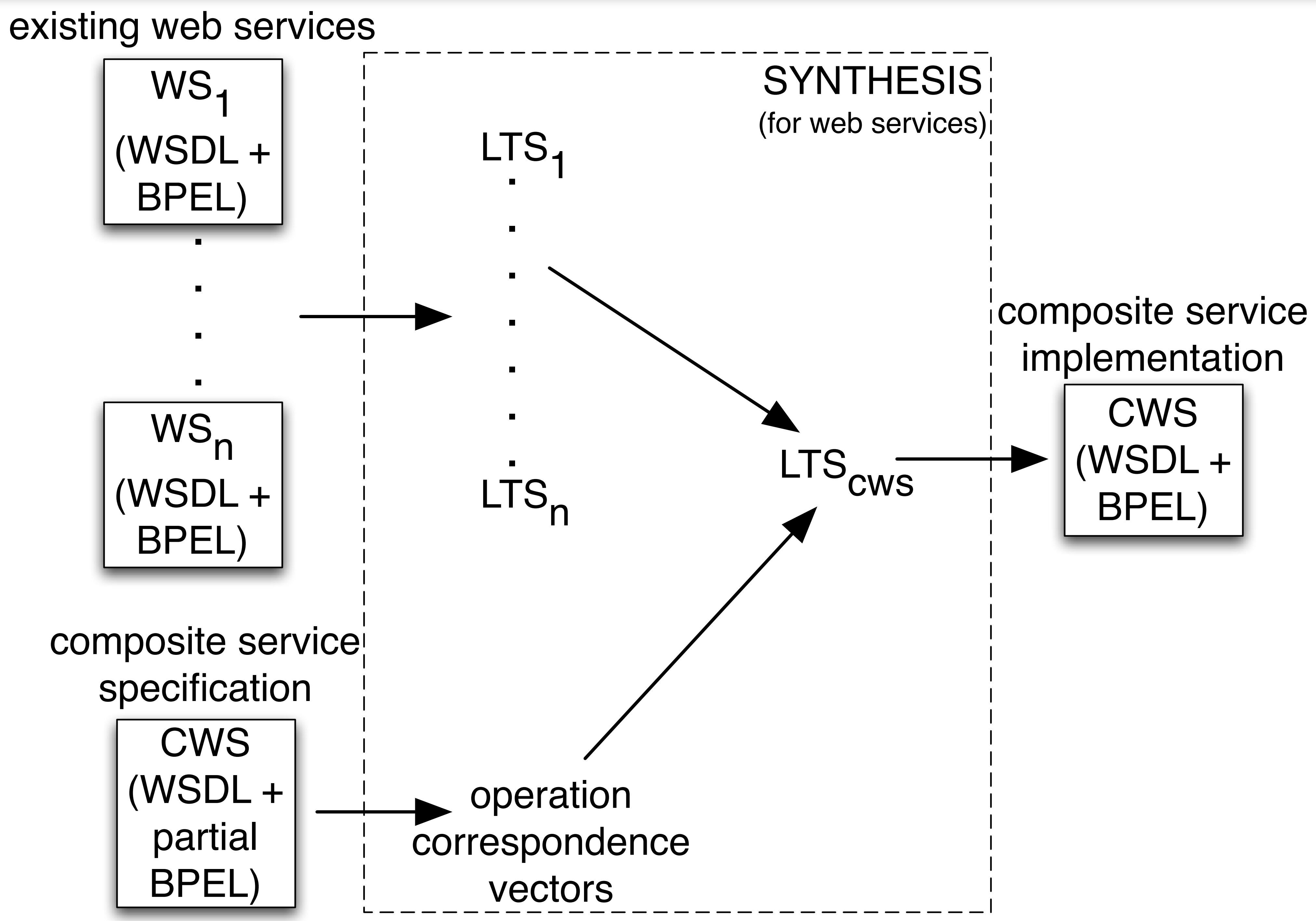
Our goal

- Automatic service composition
  - to build a new service as an automatic and correct composition of existing services
  - based on our previous work on component assembly (the *SYNTHESIS* tool)
  - web services (WSs)

Setting the context

- A centralized repository
    - e.g., UDDI registry
- Each existing WS publishes its complete SLS
    - e.g., WSDL + BPEL in the context of WSs
- The architect of the new WS specifies its partial SLS

Method's overview



Explanatory example... continuing

- 2 existing WSs: LIB and PAY
    - LIB is an old electronic library
    - PAY provides an on-line payment capability
- 1 new WS to be built: CWS
    - CWS is a new electronic library providing an on-line payment capability

Explanatory example... continuing

- **INPUT 1**: WSDL + BPEL spec. of PAY

Explanatory example... continuing

- **INPUT 1**:WSDL + BPEL spec. of PAY

WSDL

```
<definitions ...
  <portType name="PAY_PT">
    <operation name="login"> ...  </operation>
    <operation name="logout"> ...  </operation>
    <operation name="pay"> ...  </operation>
  </portType> ...
  <role name="PAY">
    <portType name="PAY_PT"/>
  </role>
  <service name="PAY_BP"/>
</definitions>
```

Explanatory example... continuing

- **INPUT 1**:WSDL + BPEL spec. of PAY

WSDL

```
<definitions ...
  <portType name="PAY_PT">
    <operation name="login"> ...  </operation>
    <operation name="logout"> ...  </operation>
    <operation name="pay"> ...  </operation>
  </portType> ...
  <role name="PAY">
    <portType name="PAY_PT"/>
  </role>
  <service name="PAY_BP"/>
</definitions>
```

Explanatory example... continuing

- **INPUT 1**: WSDL + BPEL spec. of PAY

WSDL

```
<definitions ...
  <portType name="PAY_PT">
    <operation name="login"> ...  </operation>
    <operation name="logout"> ...  </operation>
    <operation name="pay"> ...  </operation>
  </portType> ...
  <role name="PAY">
    <portType name="PAY_PT"/>
  </role>
  <service name="PAY_BP"/>
</definitions>
```

Explanatory example... continuing

- **INPUT 1**: WSDL + BPEL spec. of PAY

WSDL

```
<definitions ...
  <portType name="PAY_PT">
    <operation name="login"> ...  </operation>
    <operation name="logout"> ...  </operation>
    <operation name="pay"> ...  </operation>
  </portType> ...
  <role name="PAY">
    <portType name="PAY_PT"/>
  </role>
  <service name="PAY_BP"/>
</definitions>
```

Explanatory example... continuing

- **INPUT 1**: WSDL + BPEL spec. of PAY

WSDL

```
<definitions ...
  <portType name="PAY_PT">
    <operation name="login"> ...  </operation>
    <operation name="logout"> ...  </operation>
    <operation name="pay"> ...  </operation>
  </portType> ...
  <role name="PAY">
    <portType name="PAY_PT"/>
  </role>
  <service name="PAY_BP"/>
</definitions>
```

BPEL

```
<process name="PAY_PROCESS" ...
  <partners>
    <partner name="customer" ...  />
    <partner name="book_vendor" ...  />
  </partners> ...
  <sequence>
    <receive name="authentication" partner="customer"
      portType="PAY_PT" operation="login" .../>
    <while ...> ...
       <receive name="payment" partner="customer"
         portType="PAY_PT" operation="pay" .../>
    </while>
    <receive name="exit" partner="customer"
      portType="PAY_PT" operation="logout" .../>
  </sequence>
</process>
```

Explanatory example... continuing

- **INPUT 1**: WSDL + BPEL spec. of PAY

WSDL

```
<definitions ...
  <portType name="PAY_PT">
    <operation name="login"> ...  </operation>
    <operation name="logout"> ...  </operation>
    <operation name="pay"> ...  </operation>
  </portType> ...
  <role name="PAY">
    <portType name="PAY_PT"/>
  </role>
  <service name="PAY_BP"/>
</definitions>
```

BPEL

```
<process name="PAY_PROCESS" ...
  <partners>
    <partner name="customer" ...  />
    <partner name="book_vendor" ...  />
  </partners> ...
  <sequence>
    <receive name="authentication" partner="customer"
      portType="PAY_PT" operation="login" .../>
    <while ...> ...
       <receive name="payment" partner="customer"
         portType="PAY_PT" operation="pay" .../>
    </while>
    <receive name="exit" partner="customer"
      portType="PAY_PT" operation="logout" .../>
  </sequence>
</process>
```

Explanatory example... continuing

- **INPUT 1**: WSDL + BPEL spec. of PAY

WSDL

```
<definitions ...
  <portType name="PAY_PT">
    <operation name="login"> ...  </operation>
    <operation name="logout"> ...  </operation>
    <operation name="pay"> ...  </operation>
  </portType> ...
  <role name="PAY">
    <portType name="PAY_PT"/>
  </role>
  <service name="PAY_BP"/>
</definitions>
```

BPEL

```
<process name="PAY_PROCESS" ...
  <partners>
    <partner name="customer" ...  />
    <partner name="book_vendor" ...  />
  </partners> ...
  <sequence>
    <receive name="authentication" partner="customer"
      portType="PAY_PT" operation="login" .../>
    <while ...> ...
       <receive name="payment" partner="customer"
         portType="PAY_PT" operation="pay" .../>
    </while>
    <receive name="exit" partner="customer"
      portType="PAY_PT" operation="logout" .../>
  </sequence>
</process>
```

Explanatory example... continuing

- **INPUT 1**: WSDL + BPEL spec. of PAY

WSDL

```
<definitions ...
  <portType name="PAY_PT">
    <operation name="login"> ...  </operation>
    <operation name="logout"> ...  </operation>
    <operation name="pay"> ...  </operation>
  </portType> ...
  <role name="PAY">
    <portType name="PAY_PT"/>
  </role>
  <service name="PAY_BP"/>
</definitions>
```

BPEL

```
<process name="PAY_PROCESS" ...
  <partners>
    <partner name="customer" ...  />
    <partner name="book_vendor" ...  />
  </partners> ...
  <sequence>
    <receive name="authentication" partner="customer"
      portType="PAY_PT" operation="login" .../>
    <while ...> ...
       <receive name="payment" partner="customer"
         portType="PAY_PT" operation="pay" .../>
    </while>
    <receive name="exit" partner="customer"
      portType="PAY_PT" operation="logout" .../>
  </sequence>
</process>
```

Explanatory example... continuing

- **INPUT 1**: WSDL + BPEL spec. of PAY

WSDL

```
<definitions ...
  <portType name="PAY_PT">
    <operation name="login"> ...  </operation>
    <operation name="logout"> ...  </operation>
    <operation name="pay"> ...  </operation>
  </portType> ...
  <role name="PAY">
    <portType name="PAY_PT"/>
  </role>
  <service name="PAY_BP"/>
</definitions>
```

BPEL

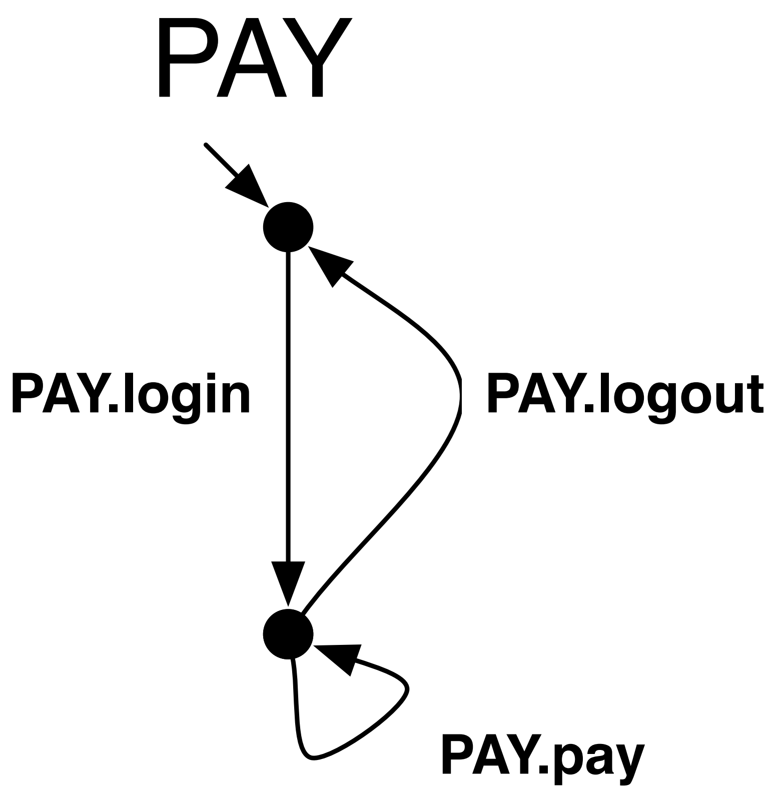
```
<process name="PAY_PROCESS" ...
  <partners>
    <partner name="customer" ...  />
    <partner name="book_vendor" ...  />
  </partners> ...
  <sequence>
    <receive name="authentication" partner="customer"
      portType="PAY_PT" operation="login" .../>
    <while ...> ...
       <receive name="payment" partner="customer"
         portType="PAY_PT" operation="pay" .../>
    </while>
    <receive name="exit" partner="customer"
      portType="PAY_PT" operation="logout" .../>
  </sequence>
</process>
```

Explanatory example... continuing

- **INPUT 1**: WSDL + BPEL spec. of PAY

$$\Big\downarrow \text{we automatically produce}$$

- **internal model 1**: LTS of PAY

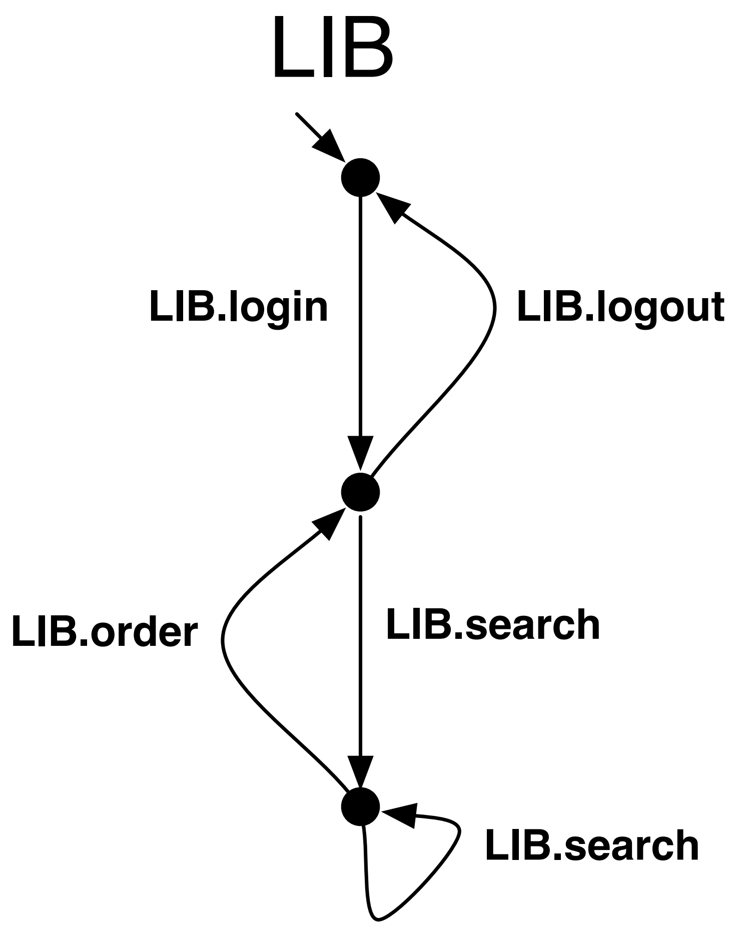


Explanatory example... continuing

- **INPUT 2**: WSDL + BPEL spec. of LIB (analogous to the spec. of PAY)

$\downarrow$ we automatically produce

- **internal model 2**: LTS of LIB



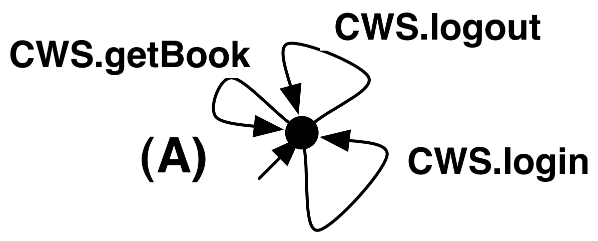
Explanatory example... continuing

- **INPUT 3**: WSDL + *partial* BPEL spec. of CWS

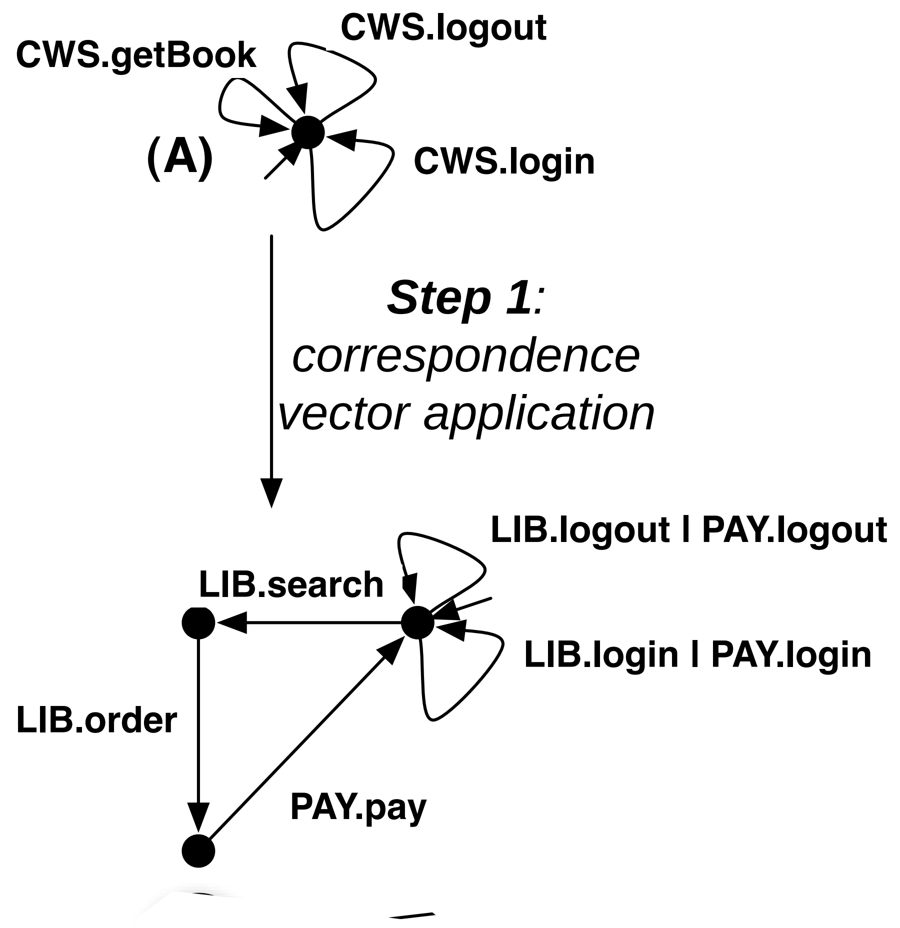
we automatically produce

parallel activities	`CWS.login ::= LIB.login | PAY.login`
parallel activities	`CWS.logout ::= LIB.logout | PAY.logout`
sequential activities	`CWS.getBook ::= LIB.search -> LIB.order -> PAY.pay`

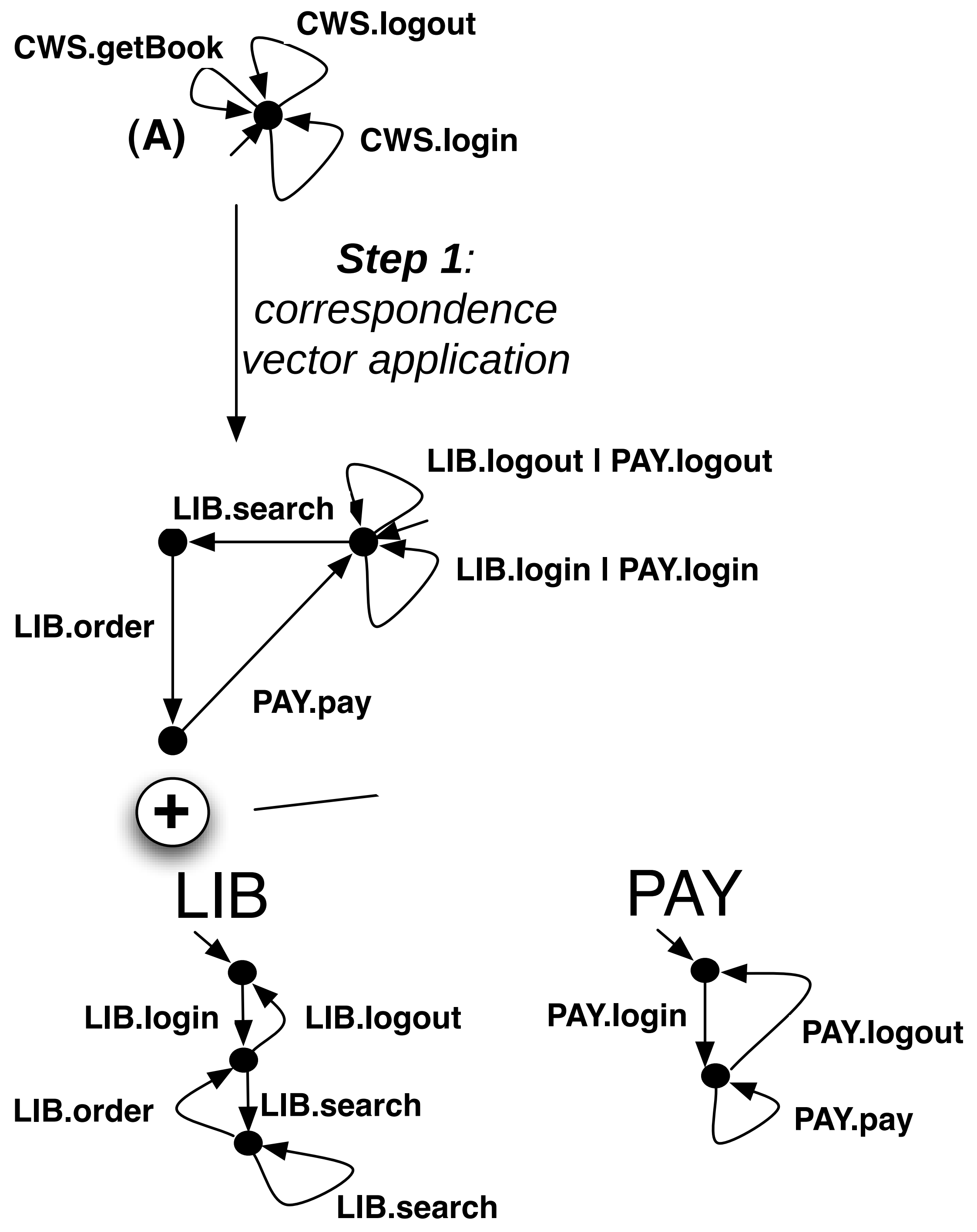
Explanatory example... concluding



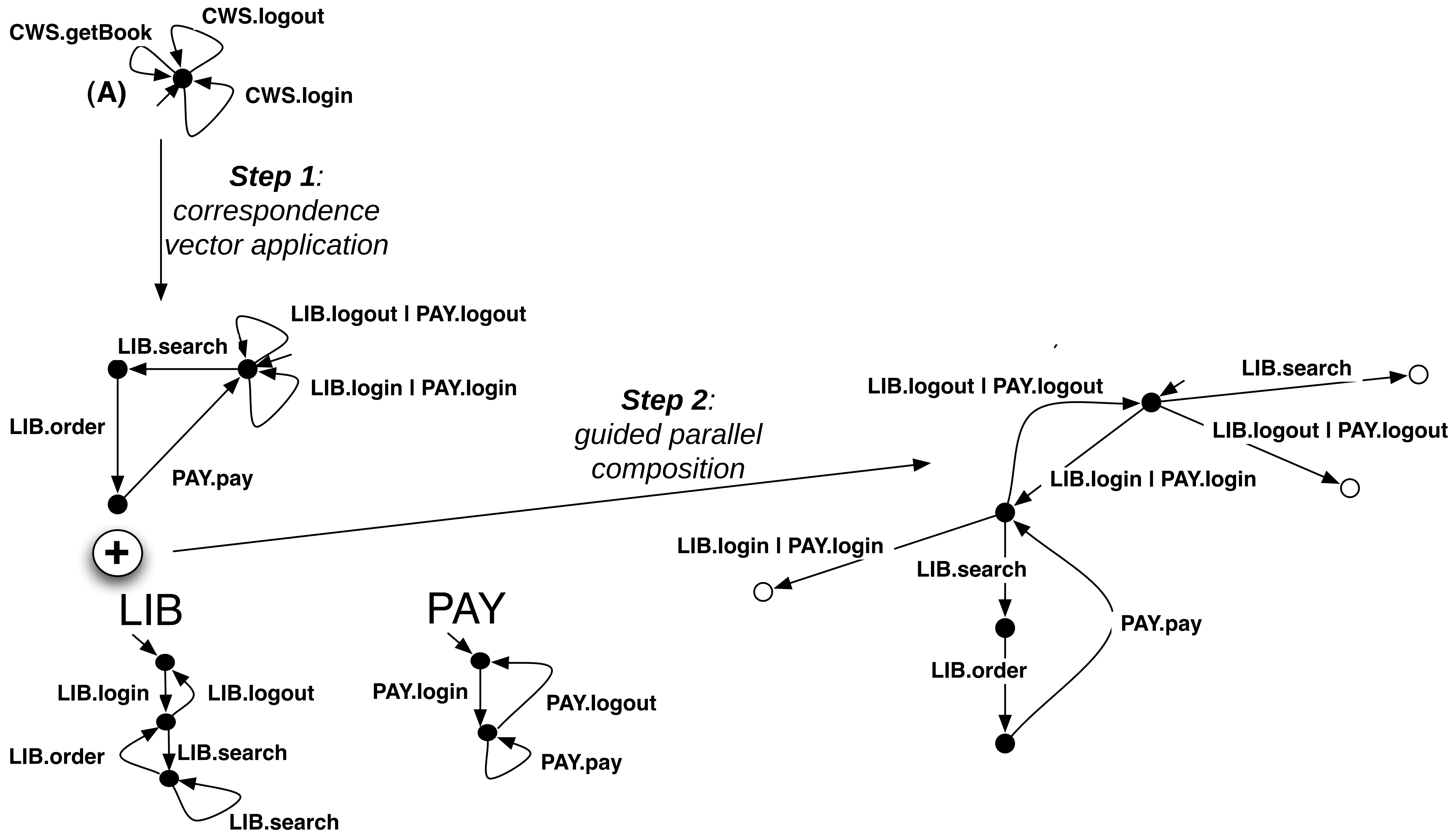
Explanatory example... concluding



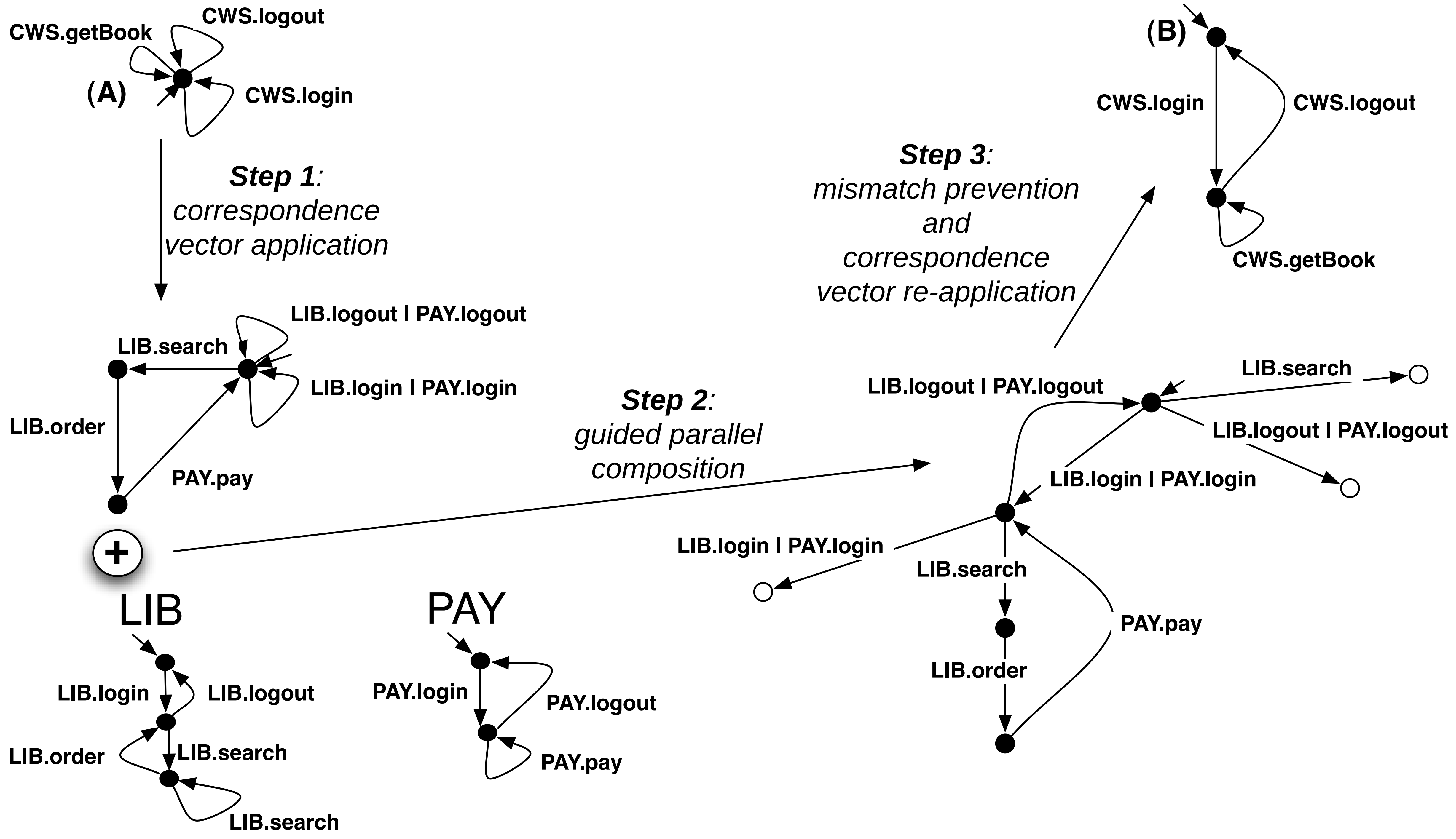
Explanatory example... concluding



Explanatory example... concluding



Explanatory example... concluding



Conclusions

- We proposed an automatic approach to the correct composition of WSs
  - i.e., automatic orchestration
- A suitable extension/modification of our SYNTHESIS tool (see the paper for details)
- The proposed approach is related to several other approaches
  - Pistore&Traverso (ASTRO project) just to mention one

Open issues

- Concerning the proposed approach
  - automatic discovery of those existing services that are "*most adequate*" for the construction of the new service
  - dealing with a more realistic SLS, i.e., with *QoS constraints*, *context-awareness*, *semantic information*, etc...
- Concerning the "*pervasive service oriented environments*" community
  - automatic service discovery
  - discovery-time orchestration or, in general, "*dynamic*" orchestration