

Towards Individualized Requirements Specification Evolution of Networked Software Based on Aspect

Zaiwen Feng, Keqing He, Yutao Ma,
Jian Wang, Ping Gong

State Key Lab of Software Engineering, Wuhan
University, China

Overview

- Background
- Patterns of requirements evolution modeling for networked software
- Individual requirements evolution inducement
- OWL-S^A
- Process of individual requirements evolution
- Conclusion

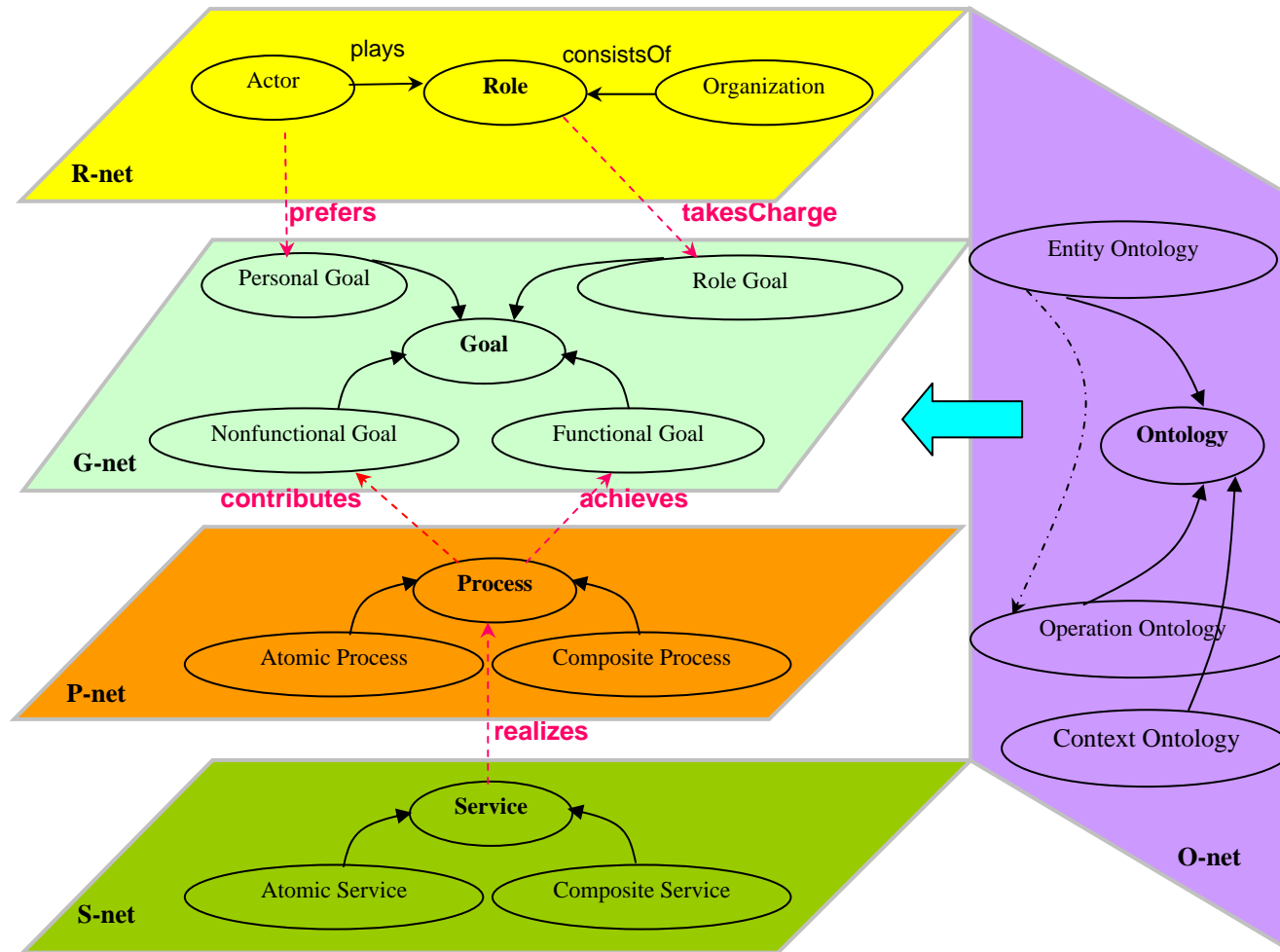
Background - Networked Software

- Definition
 - A kind of complex software systems, whose topology structure and behavior can evolve dynamically under the network environment
- Characteristics
 - Internet as the usual development and deployment platform
 - Service as the typical application form
 - Users' continuous growing and personalized requirements
 - Dynamic topology structure and evolvable software behavior

Background - RE for Networked Software

- Development Objective
 - Mass customization.
- The Way Adopted - Domain oriented and User centric
 - domain modeling provides essential reuse basis for mass customization.
 - based on the common domain knowledge, dynamic requirements evolution modeling is necessary in order to satisfy users' personalized preference, and such that the quality of experience (QoE) of users can be maximized.

O-RGPS: Meta-Models of Domain Modeling Framework for Networked Software



Requirements Evolution Modeling Based on RGPS

This is what the
paper concerns!

The evolution modeling of distributed software requirements based on RGPS framework is divided into **three** patterns:

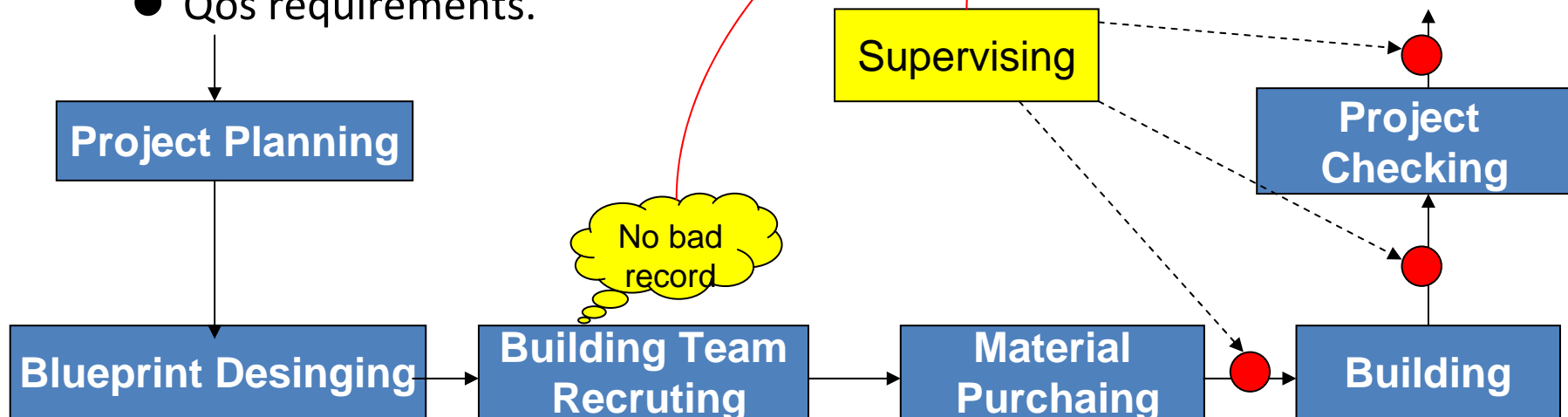
- **Weaving:** Weaving the individualized requirements specification into the original requirements specification.
- **Aggregation:** The requirements specification from different roles and different goals are aggregated to a unified system requirements specification.
- **Optimization:** The unified system requirements specification is optimized according to current context.

Inducement for individualized requirements evolution

The inducement for individualized requirements evolution is composed of:

- Additional functional requirements.
- Additional non-functional requirements.
 - business context requirements.
 - Qos requirements.

How to describes the evolutionary requirements?



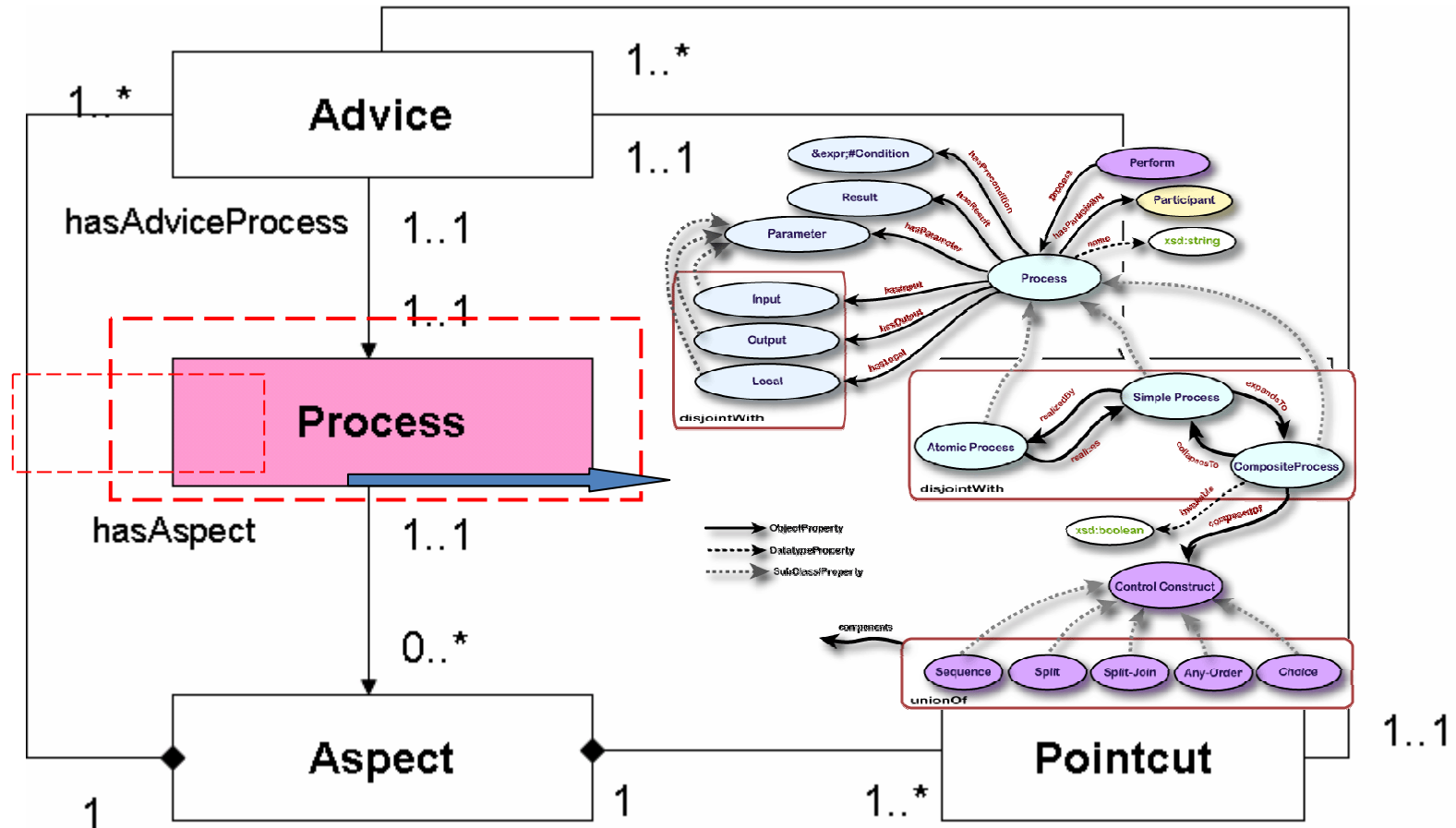
OWL-S^A: Describing the evolutionary requirements

- OWL-S can be used to describe requirements specification. But it is not suitable to describe the evolving requirements specification. It is due that OWL-S can not feature:
 - **Where** the evolution happens?
 - **When** the evolution happens?
- OWL-S^A is proposed to describe the evolutionary requirements. OWL-S^A is based on OWL-S, 'A' of OWL-S^A is refers to aspect, which comprises one or several pointcut-advice pairs.

Basic Elements of OWL-S^A

- **aspect:** An *aspect* describes an evolutionary requirements specification.
- **pointcut:** *pointcut* describes the positions where evolution takes place.
- **advice:** *advice* describes the content of requirements evolution.
- **process:** corresponding to the semantics of *process* in OWL-S.
- **evolutionaryOrder:** *evolutionaryOrder* illuminates the order between the evolutionary requirements specification and evolutionary position.

MetaModels of OWL-SA

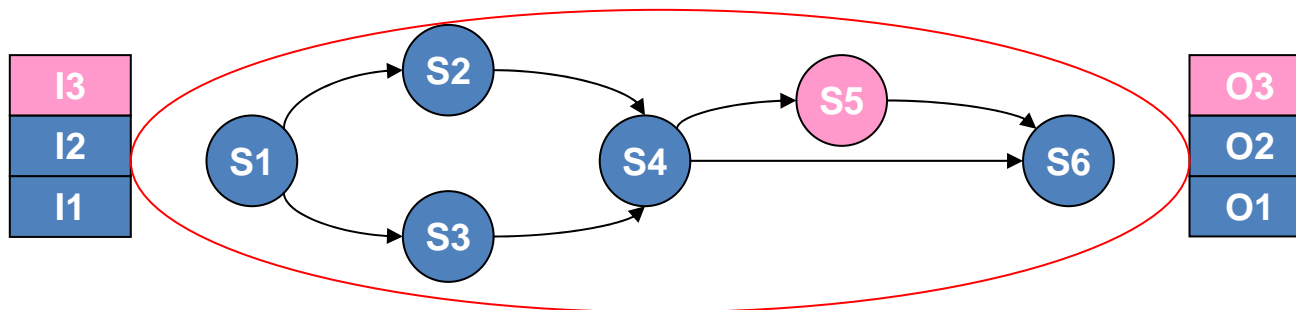


Description of evolutionary position

- Xpath is always used to address a certain node in an XML documents tree. So it can be used in OWL-S.
- *pointcut* describes the positions where evolution takes place.
- Two types of pointcuts in OWL-S^A
 - Performance to the certain process
`//process:CompositeProcess[@rdf:ID="CP"]//process:perform[@rdf:ID="STEP"]`
 - The input or output parameter of process model
`//process:CompositeProcess[@rdf:ID="CP"]//process:hasInput[@rdf:ID="I1"]`

Description of Evolutionary Content

- *advice* describes contents of requirements evolution.
- The evolutionary contents includes:
 - Imported process near pointcut.
 - Added input (output) parameter to process model.

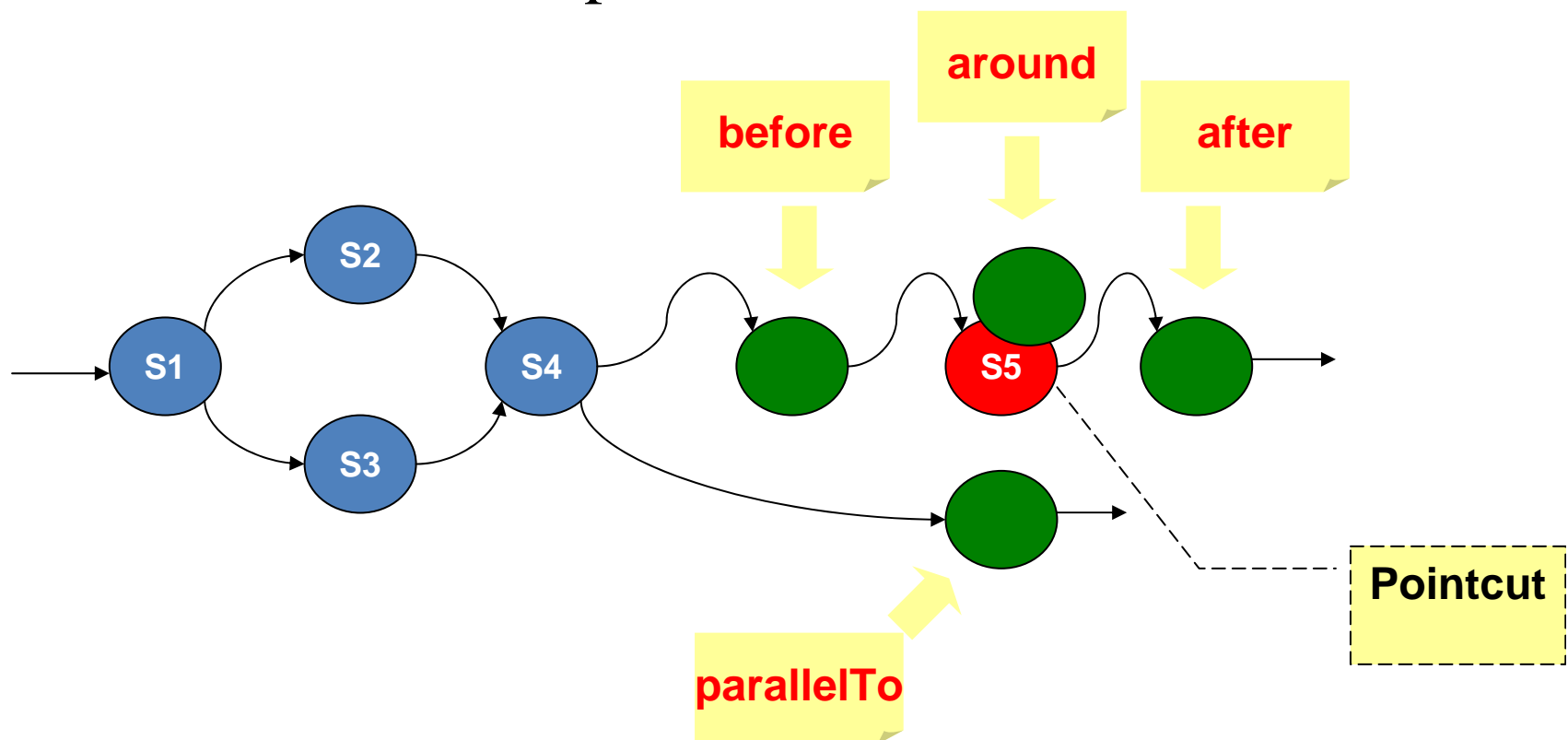


Some extensions to OWL-S

- InputBinding
 - *InputBinding* describes the data source of evolutionary process.
- ProducedBinding
 - *ProducedBinding* describes the destination of output data of evolutionary process.
- ThePointcutPerformed
 - *ThePointcutPerformed* variable represents the collection of processes that represent joint point.

Description of Evolutionary Order

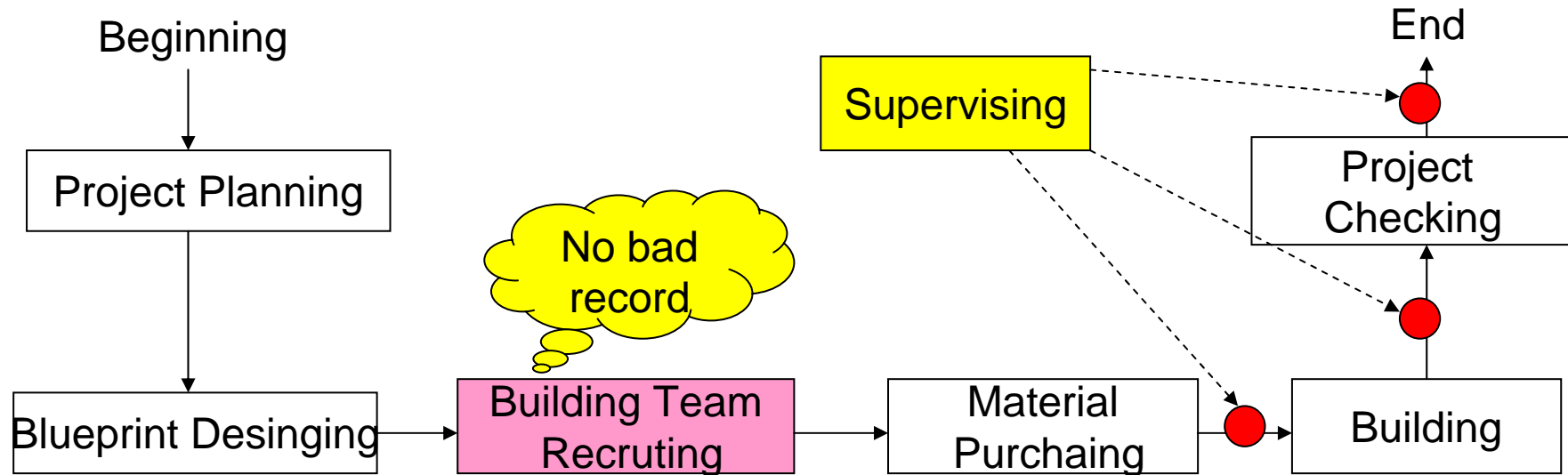
- *evolutionaryOrder* describes the sequence between evolutionary content and evolutionary position.
- four evolutionary order existed in OWL-S^A
 - before, after, around, parallelTo



Case Study

Req1: He hopes the recruited builder team has no bad record before.

Req2: User hopes the third party supervised material purchaing, building and project checking.



■ Through requirements analysis, the evolutionary requirements are composed of:

- Subsitute of old process *builder team recruiment* for new process with no bad record.
- Introduction of process *supervising* after *material purchasing*, *building* and *project checking*.

Case Study

```
<Aspect name = "Supervising">
  <pointcut name = "xxx">
    //CompositeProcess[@rdf:ID=construction]//
    Perform[@rdf:ID="Mateiral Purchasing"]
    //CompositeProcess[@rdf:ID=construction]//
    Perform[@rdf:ID="Building"]
    //CompositeProcess[@rdf:ID=construction]//
    Perform[@rdf:ID="Project Checking"]
  </pointcut>
  <advice evolutionOrder= "after">
    < Perform rdf:ID = "SupervisingPerformed">
      <process rdf:resource="&aux;#Supervising">
        <hasDataFrom>
          <InputBinding>
            <theParam rdf:resource="&aux, #I11"/>
            /*I11 is the input of Supervising*/
            <valueSource>
              < ValueOf>
                < theVar rdf:resource="#OP"/>
            /*OP refers to output of the performed pointcut.*/
            < fromProcess
              rdf:resource="#http://www.daml.org/services/o
              wl-s/1.1/Process.owl#ThePointcutPerformed"/>
            /*Setup data binding to the variable of pointcut process.*/
            <ValueOf>
              </valueSource>
            </InputBinding>
          </hasDataFrom>
```

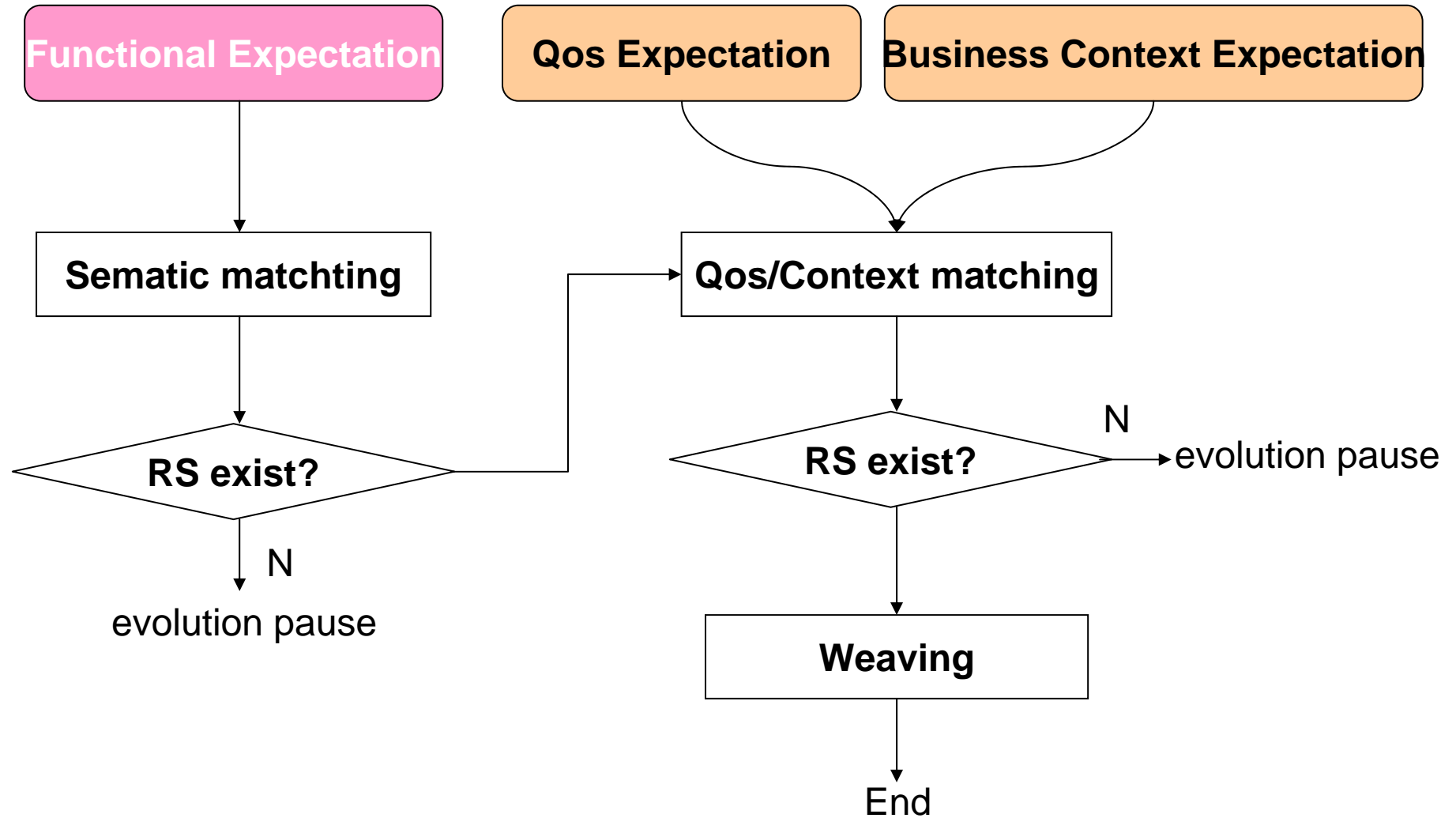
```
<Produce>
  <ProducedBinding>
    < OutputBinding>
      <theParam rdf:resource="#O11"/>
      /*O11 is the output of Supervising*/
      <valueSource>
        <ValueOf>
          <theVar rdf:resource="#O1"/>
          <fromProcessrdf:resource="#http://www.daml.or
          g/services/owls/1.1/Process.owl#TheParentPerform"/>
          /*set up data binding to output O1 of Construction*/
          <ValueOf>
            </valueSource>
          </OutputBinding>
        </ProducedBinding>
      </Produce>
    </Perform>
  </advice>
</Aspect>
```

Process of Requirements Evolution Modeling

- Step1: Discovery of process specification in domain requirements model
 - semantic matching
 - Quality matching
 - Context matching

- Step2: modification to system process model
 - Inserting
 - Substitute

Flowchart for Evolution Implementation



Conclusion

- Three patterns of requirements evolution is proposed.
- OWL-S^A which can describe the evolutionary requirements specification is defined.
- Process of individual requirements evolution modeling is proposed.

Thank you!