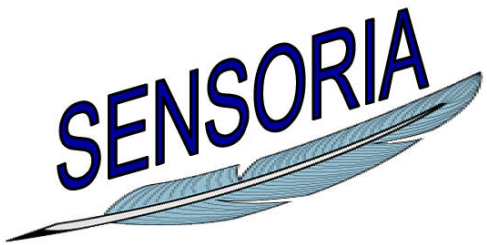
University of Leicester

Task-Oriented Business Requirements Elicitation for Web Services

Stephen Gorton

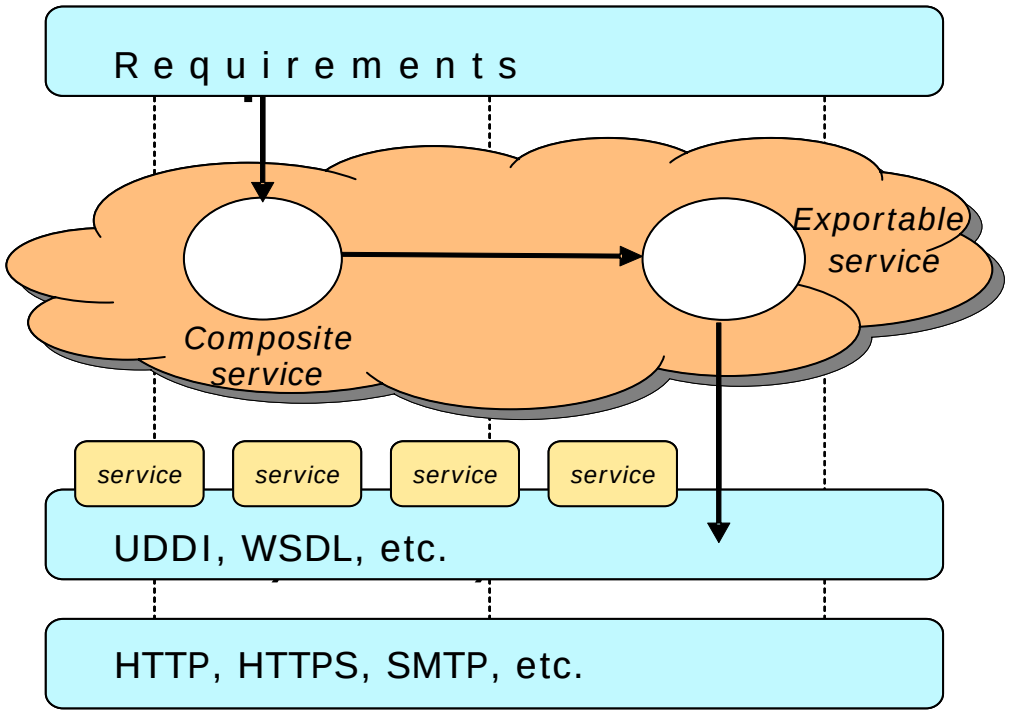


IST-FET IST-2005-16004

Department of Computer Science, University of Leicester, University Road, Leicester LE1 7RH

Web Service Architecture

- Composition often the top layer;
- Composition and orchestration still a blur!
- Little regard for a more abstract requirements layer.



Current Solutions 1

- Approach 1: Composition as Requirements
  - BPEL:

```
<process name="test">
<partnerLinks>
  <partnerLink name="client"/>
  <partnerLink name="serviceA"/>
  <partnerLink name="serviceB"/>
  <partnerLink name="serviceC"/>
</partnerLinks>
<variables>
  <variable name="processInput"/>
  <variable name="AInput"/>
  <variable name="AOutput"/>
  <variable name="BCInput"/>
  <variable name="BOutput"/>
  <variable name="COutput"/>
  <variable name="processOutput"/>
  <variable name="AError"/>
</variables>
<sequence>
  <receive name="receiveInput"
    variable="input"/>
    <assign>
      <copy>
        <from variable="processInput"/>
        <to variable="AInput"/>
      </copy>
</assign>
<scope>
  <faultHandlers>
    <catch
      faultName="faultA"
      fault-
      Variable="AError"/
      >
  </faultHandlers>
  <sequence>
    <invoke
      name="invokeA"
      partner-
      Link="serviceA"
      inputVariable="AIn
      put" output-
      Variable="AOutput"
      />
  </sequence>
</scope>
<assign>
  <copy>
    <from
    variable="AOutput"/>
      <to
      variable="BCInput"/>
    </copy>
</assign>
<flow>
  <sequence>
    <invoke
      name="invokeB"
      partner-
      Link="serviceB"
      inputVariable="BCI
      nput"/>
    <receive
      name="receive_invo
      keB"
      partnerLink="servi
      ceB"
      variable="BOutput"
      />
  </sequence>
  <sequence>
    <invoke
      name="invokeC"
      partner-
      Link="serviceC"
      inputVariable="BCI
      nput"/>
    <receive
      name="receive_invo
      keC"
      partnerLink="servi
      ceC"
      variable="COutput"
      />
  </sequence>
</flow>
<switch>
  <case>
    <!-assign value to
    processOutput->
  </case>
</switch>
<invoke name="reply"
  partnerLink="client"
  inputVariable="process
  Output"/>
</sequence>
</process>
```

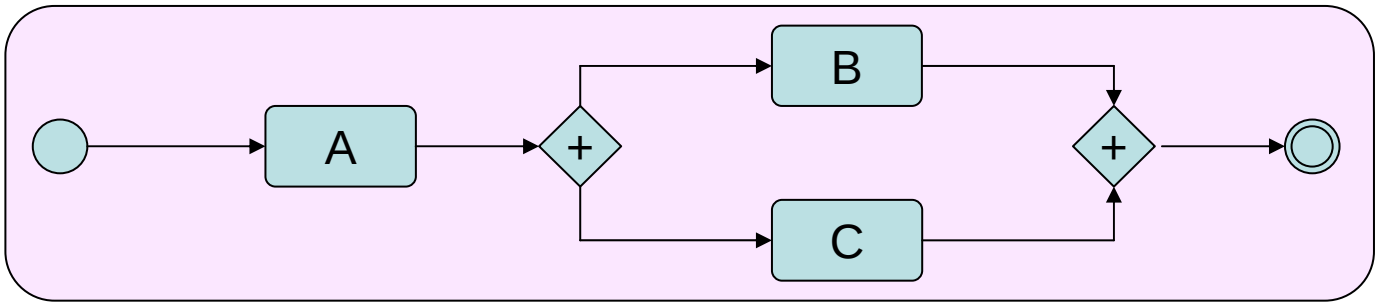
  - DAML-S

```
<daml:Class rdf:ID="test">
 <daml:subClassOf
rdf:resource="Process.CompositeProcess"/>
 <daml:subClassOf>
  <daml:Restriction>
   <daml:onProperty
rdf:resource="Process#composedOf"/>
   <daml:toClass>
    <daml:Class>
     <daml:intersectionOf rdf:parse-
Type="daml:collection">
      <daml:Class
rdf:about="process:Sequence">
       <daml:Restriction>
        <daml:onProperty
rdf:resource="Process#components"/>
        <daml:toClass>
         <daml:Class>
          <process:listOfInstancesOf
              rdf:parseType="daml:col-
lection">
          <daml:Class rdf:about="#ser-
viceA"/>
          <daml:Class
rdf:about="process:Split">
           <daml:Restriction>
            <daml:onProperty
rdf:resource="Process#components"/>
            <daml:toClass>
             <daml:Class>
              <process:listOfInstancesOf
                 rdf:parseType="daml:col-
lection">
               <daml:Class
rdf:about="#serviceB"/>
               <daml:Class
rdf:about="#serviceC"/>
              </process:listOfInstance-
sOf>
             </daml:Class>
          . . .
```

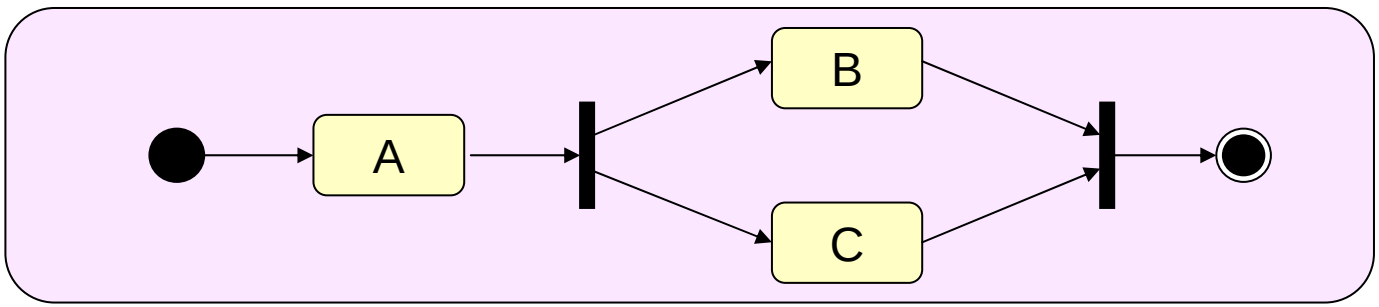
Code snippets taken from Milanovic and Malek: Current Solutions for Web Service Composition. IEEE Internet Computing, Nov/Dec 04

Current Solutions 2

- Approach 2: Specialised Requirements Language
  - BPMN:



  - UML:



Wedding Example

- Business goal *g* = “plan wedding”;
- Broken down into objectives (composite tasks):
  - ct_1 = plan pre-wedding celebrations;
  - ct_2 = plan preparations;
  - ct_3 = plan legalities;
  - ct_4 = plan ceremony;
  - ct_5 = plan post-ceremony celebrations;
  - ct_6 = plan honeymoon.
- Tasks are arranged according to result timeline, not according to execution timeline!
  - e.g. ceremony and post-ceremony celebrations often planned in parallel.
- Policies:
  - The entire event should not cost more than £10k;
  - The ceremony and post-ceremony celebrations should be on the same day;
  - The honeymoon should be booked through a known and trusted travel agency.

Booking the Honeymoon 1

Flows:

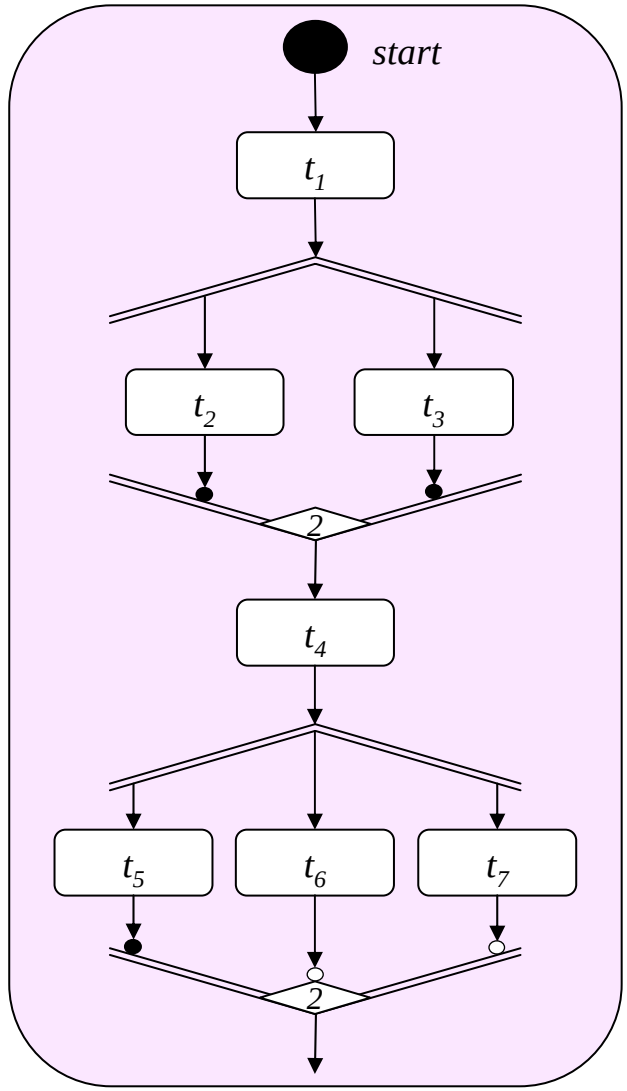
- Control runs from start to finish;
- Solid lines indicate control flow routes;
- A task is executed when control reaches it;
- Control proceeds when the task has finished.

Flow Split:

- FS: *in -> OUT*;
- Control proceeds down each output simultaneously;
- No limit on number of output flows;
- Parallel split workflow pattern

Conditional Merge:

- CM: *IN -> out*;
- Forces synchronisation;
- Mandatory and optional flows;
- Specifies minimum number of flows;
- Discriminator workflow pattern.



Booking the Honeymoon 2

Strict Preference:

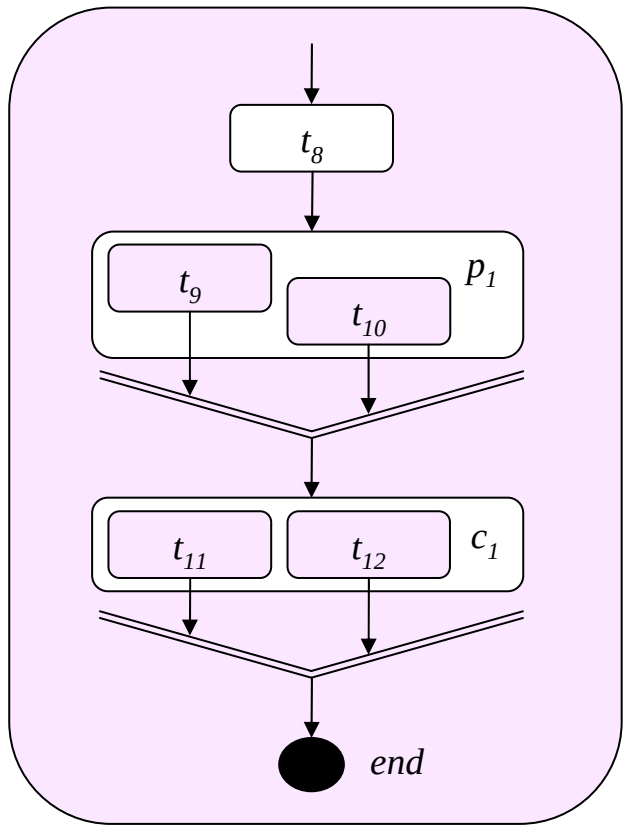
- SP: *in -> out*;
- Input is a set of pairs {*t, n*}
  - *t* is a task;
  - *n* is a priority rating;
- New workflow pattern.

Flow Merge:

- FM: *IN -> out*;
- Incoming set of control flows contains only one active flow;
- No synchronisation issue;
- (Multiple) Merge workflow pattern.

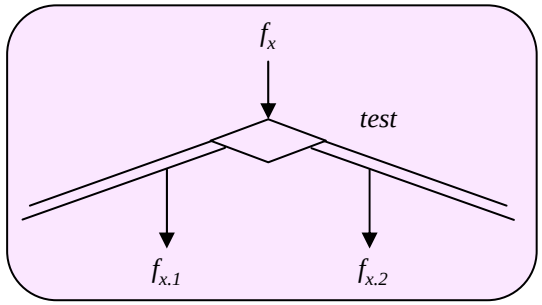
Random Choice:

- RC: *in -> out*;
- All tasks invoked;
- When a first gets to a “commit”, all others are cancelled;
- New workflow pattern.

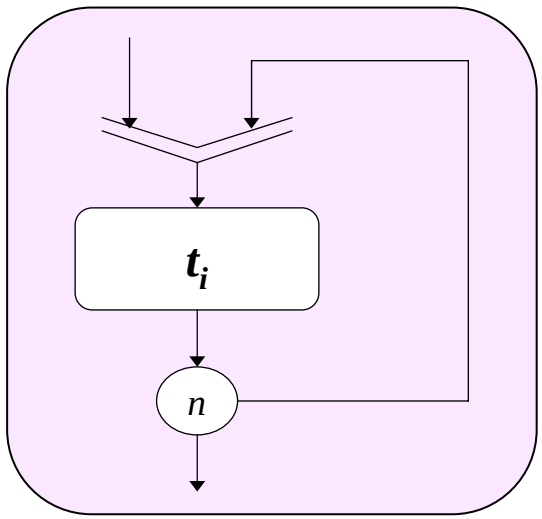


Other Notation

- Flow Junction Operator:
  - FJ: *in* -> {out_1, out_2};
  - Left output is primary;
  - Output flow chosen according to a test;
  - Exclusive choice workflow pattern.



- Bounded cycles allowed:
  - For both composite and atomic tasks;
  - Can be modelled with flow junction and flow merge.
  - (since we only allow one control flow input, a flow merge function should be used).



Summary

- Current notations not appropriate:
  - UML has some merits but does not support many workflow patterns;
  - BPMN is the nearest to a complete solution;
  - None allow for the expression of all requirements.

- A simple graphical notation:
  - Describing process flows;
  - Scope for core and non-core (non-functional) requirements;
  - Offers the context in which policies are used.

- Further work:
  - Workflow patterns (data and resource patterns);
  - Policies and policy framework at the business level;
  - A workbench.

Thank you.

Any Questions?