

*Web Service Composition Software with the  
Visual User Interface*

*(Visuaalse kasutajaliidesega veebiteenuste  
kompositsioonitarkvara)*

Riina Maigre

Institute of Cybernetics at TUT

Põlva 2008

# *Outline*

## *Introduction*

- Web services

- Service composition

## *Higher order workflows (HOWF)*

- HOWF and representation in logic

- Proof of solution

## *Service composition in CoCoViLa*

- CoCoViLa

- Composition steps

## *Example*

- X-road data exchange layer

- X-road service model

- X-road service model in CoCoViLa

## *Web services*

Web service is software component that accessible over the Web.



Web service description includes:

- ▶ name, description, inputs and outputs
- ▶ location
- ▶ invocation information

Service description languages: WSDL, WSDL-S, SA-WSDL, WSMO, OWL-S, ...

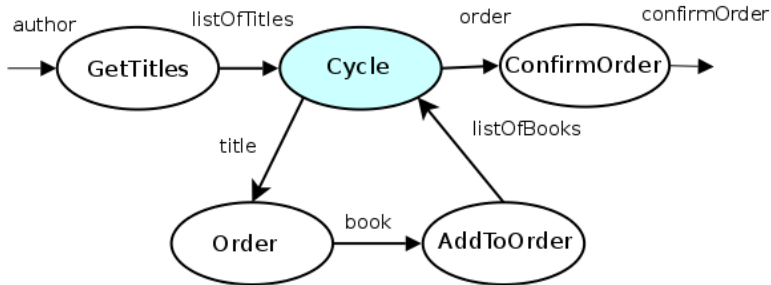
## *Web service composition*

- ▶ Web service composition is the task of combining and linking existing Web services to create new Web processes in order to add value to the collection of services.
- ▶ Web service composition can include services from several different computers.
- ▶ Composition constructions can be sequences, cycles, conditions etc.
- ▶ Composition languages: WS-BPEL, WSMO, OWL-S, XLANG, WSFL, BPML, WSCI, ...

## *Problems*

- ▶ Expressing the meaning of composed service is complicated from the user's perspective (Petri nets, state-machines),
- ▶ Restricted visualisation possibilities (Protégé OWL-S editor , OWL-S IDE, Eclipse BPEL designer etc),
- ▶ Automatic composition needs precise semantics.

## Higher order workflows and representation in logic



$\text{author} \supset \text{listOfTitles} : \text{GetTitles} \quad (\text{T1})$

$\text{order} \supset \text{confirmOrder} : \text{ConfirmOrder} \quad (\text{T2})$

$\text{title} \supset \text{book} : \text{Order} \quad (\text{T3})$

$\text{book} \supset \text{listOfBooks} : \text{AddToOrder} \quad (\text{T4})$

$\text{listOfTitles} \wedge (\text{title} \supset \text{listOfBooks}) \supset \text{order} : \text{Cycle} \quad (\text{T5})$

$\text{author} \supset \text{confirmOrder}$

# Proof of solution

## Formulae:

author  $\supset$  listOfTitles : *GetTitles* (T1)

order  $\supset$  confirmOrder : *ConfirmOrder* (T2)

title  $\supset$  book : *Order* (T3)

book  $\supset$  listOfBooks : *AddToOrder* (T4)

listOfTitles  $\wedge$  (title  $\supset$  listOfBooks)  $\supset$  order : *Cycle*

## Rules:

$$\frac{A \supset B \wedge C : f \quad B \wedge D \supset G : g}{A \wedge D \supset C \wedge G \wedge B : f; g} \quad (1)$$

$$\frac{(A \supset B) \wedge X \supset Z : f \quad A \wedge W \supset B : g}{X \vee W \supset Z : f(g)} \quad (2)$$

author  $\supset$  confirmOrder

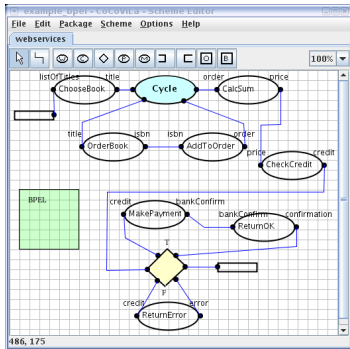
## Proof:

$$\frac{\frac{\frac{\text{T3} \quad \text{book} \supset \text{listOfBooks} : \textit{AddToOrder}}{\text{title} \supset \text{listOfBooks} : \textit{Order}; \textit{AddToOrder}} \quad (1) \quad \text{T5}}{\text{listOfTitles} \supset \text{order} : \textit{Cycle}(\textit{Order}; \textit{AddToOrder})} \quad (2) \quad \text{T2}}{\text{T1} \quad \text{listOfTitles} \supset \text{confirmOrder} : \textit{Cycle}(\textit{Order}; \textit{AddToOrder}); \textit{ConfirmOrder}} \quad (1)$$
$$\text{author} \supset \text{confirmOrder} : \textit{GetTitles}; \textit{Cycle}(\textit{Order}; \textit{AddToOrder}); \textit{ConfirmOrder} \quad (1)$$

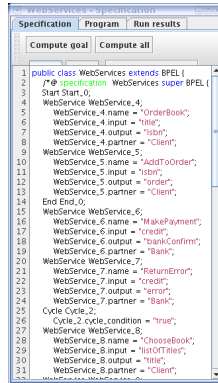
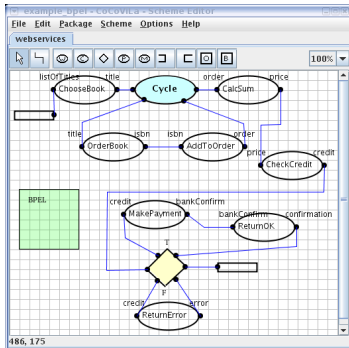
## *Service composition in CoCoViLa*

- ▶ CoCoViLa provides a framework for developing visual specification languages.
- ▶ Includes algorithm synthesis based on Structural Synthesis of Programs.
- ▶ Composition steps:
  - ▶ Specification
  - ▶ Problem description in logic and formulation of the goal
  - ▶ Proof of solvability of the problem
  - ▶ Java program

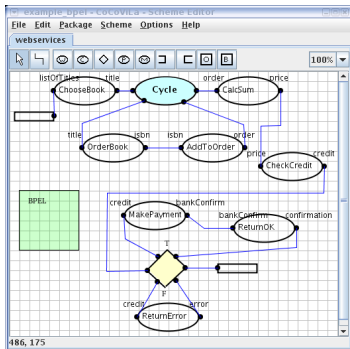
# Composition steps



# Composition steps



# Composition steps



WebServices - Specification

Specification Program Run results

Compute goal Compute all

```

1 public class WebServices extends BPEL {
2   /* @ specification WebServices super BPEL {
3   Start Start; 0;
4   WebService WebService_4;
5   WebService_4.name = "OrderBook";
6   WebService_4.input = "title";
7   WebService_4.output = "isbn";
8   WebService_4.partner = "Client";
9   WebService WebService_5;
10  WebService_5.name = "AddToOrder";
11  WebService_5.input = "isbn";
12  WebService_5.output = "order";
13  WebService_5.partner = "Client";
14  End End; 0;
15  WebService WebService_6;
16  WebService_6.name = "MakePayment";
17  WebService_6.input = "credit";
18  WebService_6.output = "bankConfirm";
19  WebService_6.partner = "Bank";
20  WebService WebService_7;
21  WebService_7.name = "ReturnError";
22  WebService_7.input = "credit";
23  WebService_7.output = "error";
24  WebService_7.partner = "Bank";
25  Cycle Cycle_2;
26  Cycle_2.cycle_condition = "true";
27  WebService WebService_8;
28  WebService_8.name = "ChooseBook";
29  WebService_8.input = "listOfTitles";
30  WebService_8.output = "title";
31  WebService_8.partner = "Client";
32  }
  
```

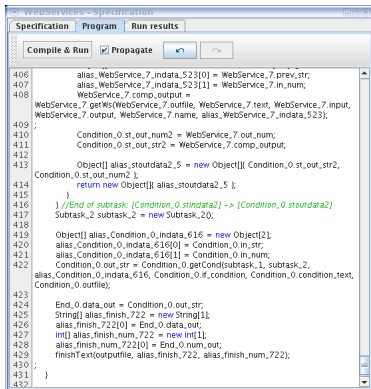
Algorithm visualizer

WebServices

```

spec : alias (int) finish_num = (* num_out)
Cycle_2 : [st_indata -> st_outdata], cycle_condition, cycle_indata, cycle_text, outfile, prev_output -> cycle_outdata_str (getCycle)
  Subtask [st_indata -> st_outdata] :
    spec : WebService_4.indata = Cycle_2.st_indata
    WebService_4 : alias indata = (prev_str, in_num)
    WebService_4 : out_num = in_num + 1
    WebService_4 : outfile, text, input, output, name, indata -> comp_output (getWs)
    spec : WebService_5.indata = WebService_4.outdata
    WebService_4 : alias outdata = (comp_output, out_num)
    spec : WebService_5.indata = WebService_4.outdata
    WebService_5 : out_num = in_num + 1
    WebService_5 : alias indata = (prev_str, in_num)
    WebService_5 : prev_output = prev_str
    spec : Cycle_2.st_outdata = WebService_5.outdata
    WebService_5 : outfile, text, input, output, name, indata -> comp_output (getWs)
    WebService_5 : alias outdata = (comp_output, out_num)
    spec : Cycle_2.st_outdata = WebService_5.outdata
    Cycle_2 : alias st_outdata = (st_outdata_str, st_outdata_num)
    Cycle_2 : prev_stoutput = st_outdata_str
  End of : Cycle_2 : [st_indata -> st_outdata], cycle_condition, cycle_indata, cycle_text, outfile, prev_output -> cycle_outdata_str (getCycle)
  
```

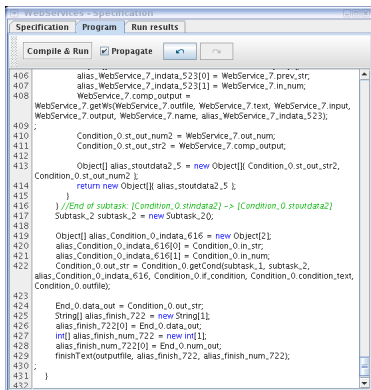
# Composition steps



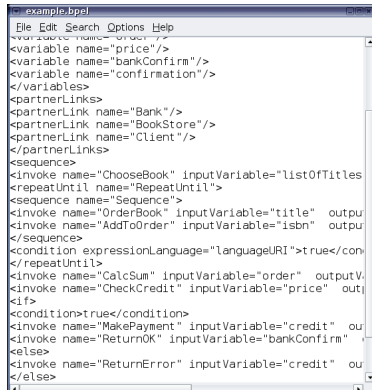
The screenshot shows a software interface titled "WebServices - Specification". It has three tabs: "Specification", "Program", and "Run results". The "Program" tab is active, displaying a Java code snippet. Above the code, there are buttons for "Compile & Run" and "Propagate" (which is checked). The code is a Java program that manipulates arrays and objects, likely representing web service data. It includes comments and uses various Java constructs like arrays, objects, and conditional logic.

```
406     alias_WebService_7_indata_523[0] = WebService_7_prev_str;  
407     alias_WebService_7_indata_523[1] = WebService_7_in_num;  
408     WebService_7_comp_output =  
WebService_7_getWs(WebService_7_outfile, WebService_7_text, WebService_7_input,  
WebService_7_output, WebService_7_name, alias_WebService_7_indata_523);  
409  
410     Condition_0_st_out_num2 = WebService_7_out_num;  
411     Condition_0_st_out_str2 = WebService_7_comp_output;  
412  
413     Object[] alias_stoutdata2_5 = new Object[] { Condition_0_st_out_str2,  
Condition_0_st_out_num2 };  
414     return new Object[] { alias_stoutdata2_5 };  
415 }  
416 } //End of subtask: [Condition_0_stindata2] -> [Condition_0_stoutdata2]  
417 Subtask_2 subtask_2 = new Subtask_2();  
418  
419 Object[] alias_Condition_0_indata_616 = new Object[2];  
420 alias_Condition_0_indata_616[0] = Condition_0_in_str;  
421 alias_Condition_0_indata_616[1] = Condition_0_in_num;  
422 Condition_0_out_str = Condition_0.getCond(subtask_1, subtask_2,  
alias_Condition_0_indata_616, Condition_0.if_condition, Condition_0.condition_text,  
Condition_0_outfile);  
423  
424 End_0_data_out = Condition_0_out_str;  
425 String[] alias_finish_722 = new String[1];  
426 alias_finish_722[0] = End_0_data_out;  
427 int[] alias_finish_num_722 = new int[1];  
428 alias_finish_num_722[0] = End_0_num_out;  
429 finishText(outputfile, alias_finish_722, alias_finish_num_722);  
430  
431 ;  
432 }
```

# Composition steps



```
Specification Program Run results
Compile & Run Propagate
406 alias_WebService_7_indata_523[0] = WebService_7_prev_str;
407 alias_WebService_7_indata_523[1] = WebService_7_in_num;
408 WebService_7_comp_output =
WebService_7_getWs(WebService_7_outfile, WebService_7_text, WebService_7_input,
WebService_7_output, WebService_7_name, alias_WebService_7_indata_523);
409
410 Condition_0_st_out_num2 = WebService_7_out_num;
411 Condition_0_st_out_str2 = WebService_7_comp_output;
412
413 Object[] alias_stoutdata2_5 = new Object[] { Condition_0_st_out_str2,
Condition_0_st_out_num2 };
414 return new Object[] { alias_stoutdata2_5 };
415
416 } //End of Subtask: [Condition_0_stoutdata2] -> [Condition_0_stoutdata2]
417 Subtask_2 subtask_2 = new Subtask_2();
418
419 Object[] alias_Condition_0_indata_616 = new Object[2];
420 alias_Condition_0_indata_616[0] = Condition_0_in_str;
421 alias_Condition_0_indata_616[1] = Condition_0_in_num;
422 Condition_0_out_str = Condition_0_getCond(subtask_1, subtask_2,
alias_Condition_0_indata_616, Condition_0_if_condition, Condition_0_condition_text,
Condition_0_outfile);
423
424 End_0_data_out = Condition_0_out_str;
425 String[] alias_finish_722 = new String[1];
426 alias_finish_722[0] = End_0_data_out;
427 int[] alias_finish_num_722 = new int[1];
428 alias_finish_num_722[0] = End_0_num_out;
429 finishText(outputfile, alias_finish_722, alias_finish_num_722);
430 ;
431 }
432
```



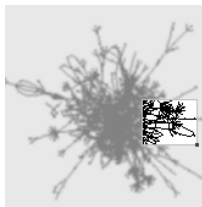
```
example.bpel
File Edit Search Options Help
<variable name="price"/>
<variable name="bankConfirm"/>
<variable name="confirmation"/>
</variables>
<partnerLinks>
<partnerLink name="Bank"/>
<partnerLink name="BookStore"/>
<partnerLink name="Client"/>
</partnerLinks>
<sequence>
<invoke name="ChooseBook" inputVariable="listOfTitles"
<repeatUntil name="RepeatUntil">
<sequence name="Sequence">
<invoke name="OrderBook" inputVariable="title" output
<invoke name="AddToOrder" inputVariable="isbn" output
</sequence>
<condition expressionLanguage="languageURI">true</condition>
</repeatUntil>
<invoke name="CalcSum" inputVariable="order" outputV
<invoke name="CheckCredit" inputVariable="price" out
<if>
<condition>true</condition>
<invoke name="MakePayment" inputVariable="credit" ou
<invoke name="ReturnOK" inputVariable="bankConfirm"
<else>
<invoke name="ReturnError" inputVariable="credit" ou
</else>

```

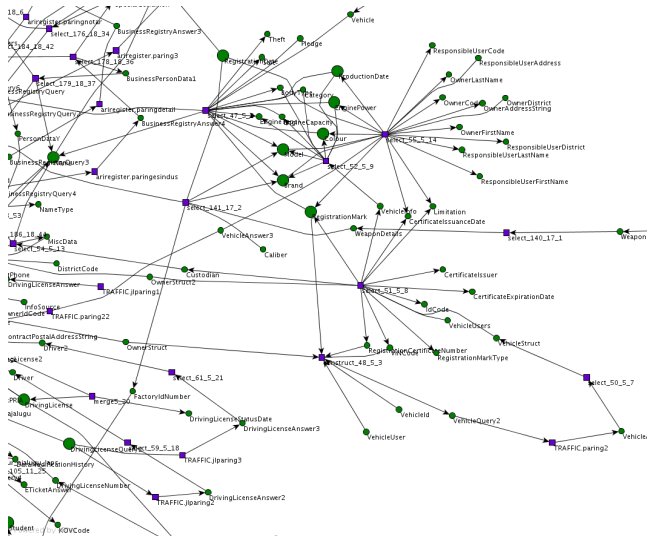
## *X-road data exchange layer*

- ▶ X-road enables secure access to nearly all Estonian national databases.
- ▶ Hundreds of services provided by information systems of different institutions work over the X-Road on the 24/7 basis.
- ▶ All Estonian residents with the national ID card or a contract for the use of Internet banking codes can make use of its enquiry services targeted at citizens.
- ▶ Officials as well as legal and natural persons are allowed to search data from national databases over the Internet within the limits of their authority.

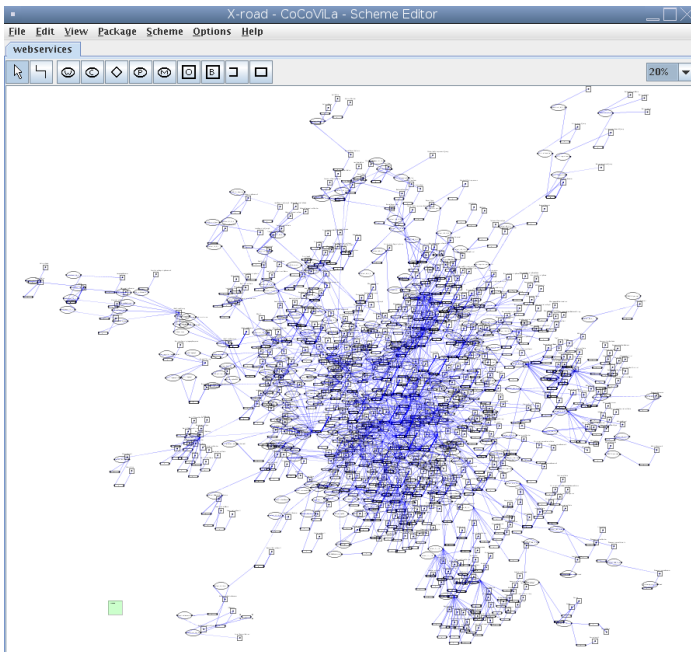
### *X-road model*



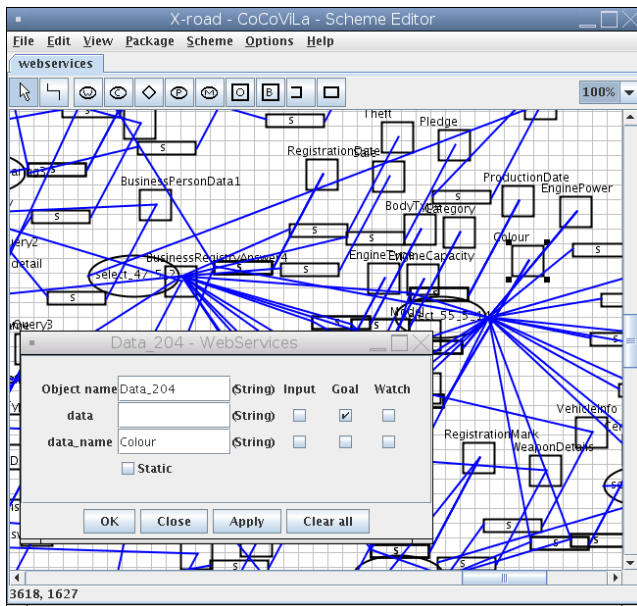
(Peep Kūngas)



# *X-Road services model in CoCoViLa*



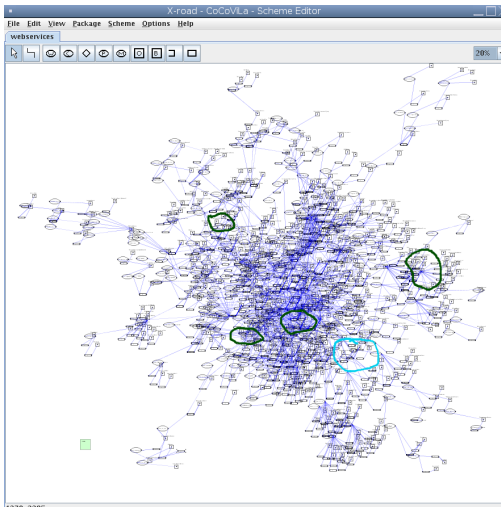
## *X-Road services in CoCoViLa*



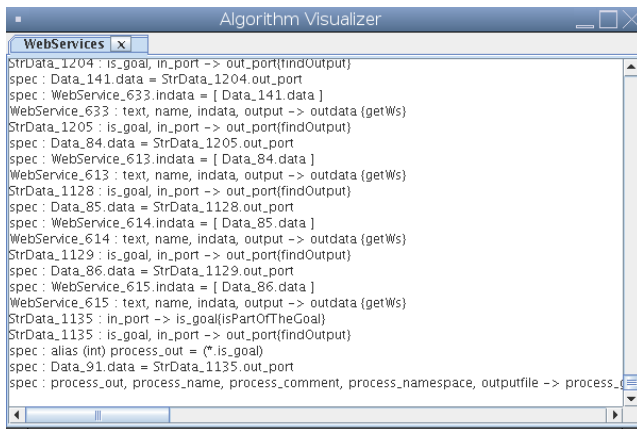
# *X-Road services in CoCoViLa*

GraduationCertificate— >

RealEstateType, RegistrationMark, Colour, EstonianAddressString,  
OccupationArea



## *X-Road services in CoCoViLa*

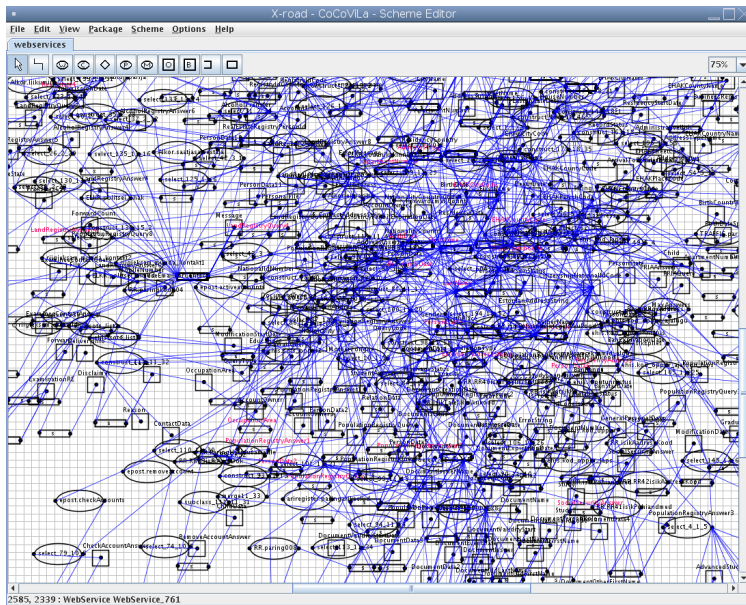


The screenshot shows a window titled "Algorithm Visualizer" with a tab labeled "WebServices". The window contains a list of web services and their specifications, including data flows and process names.

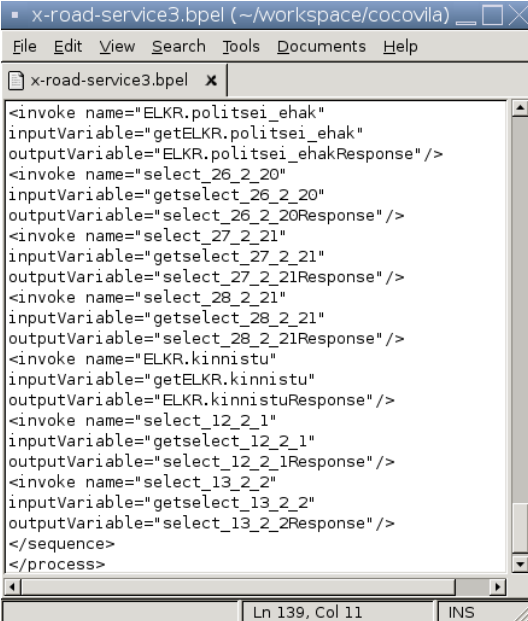
```
StrData_1204 : is_goal, in_port -> out_port(findOutput)
spec : Data_141.data = StrData_1204.out_port
spec : WebService_633.indata = [ Data_141.data ]
WebService_633 : text, name, indata, output -> outdata {getWs}
StrData_1205 : is_goal, in_port -> out_port(findOutput)
spec : Data_84.data = StrData_1205.out_port
spec : WebService_613.indata = [ Data_84.data ]
WebService_613 : text, name, indata, output -> outdata {getWs}
StrData_1128 : is_goal, in_port -> out_port(findOutput)
spec : Data_85.data = StrData_1128.out_port
spec : WebService_614.indata = [ Data_85.data ]
WebService_614 : text, name, indata, output -> outdata {getWs}
StrData_1129 : is_goal, in_port -> out_port(findOutput)
spec : Data_86.data = StrData_1129.out_port
spec : WebService_615.indata = [ Data_86.data ]
WebService_615 : text, name, indata, output -> outdata {getWs}
StrData_1135 : in_port -> is_goal(isPartOfTheGoal)
StrData_1135 : is_goal, in_port -> out_port(findOutput)
spec : alias (int) process_out = (*.is_goal)
spec : Data_91.data = StrData_1135.out_port
spec : process_out, process_name, process.comment, process_namespace, outputfile -> process_out
```

Finding the solution takes about one second.

# X-Road services in CoCoViLa



## *X-Road services in CoCoViLa*



The screenshot shows a BPEL editor window titled "x-road-service3.bpel (~/.workspace/cocovila)". The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. The editor displays a BPEL process with the following code:

```
<invoke name="ELKR.politsei_ehak"
inputVariable="getELKR.politsei_ehak"
outputVariable="ELKR.politsei_ehakResponse"/>
<invoke name="select_26_2_20"
inputVariable="getselect_26_2_20"
outputVariable="select_26_2_20Response"/>
<invoke name="select_27_2_21"
inputVariable="getselect_27_2_21"
outputVariable="select_27_2_21Response"/>
<invoke name="select_28_2_21"
inputVariable="getselect_28_2_21"
outputVariable="select_28_2_21Response"/>
<invoke name="ELKR.kinnistu"
inputVariable="getELKR.kinnistu"
outputVariable="ELKR.kinnistuResponse"/>
<invoke name="select_12_2_1"
inputVariable="getselect_12_2_1"
outputVariable="select_12_2_1Response"/>
<invoke name="select_13_2_2"
inputVariable="getselect_13_2_2"
outputVariable="select_13_2_2Response"/>
</sequence>
</process>
```

The status bar at the bottom indicates "Ln 139, Col 11" and "INS" mode.

# *Conclusions*

The proposed method:

- ▶ allows to design composition visually,
- ▶ is suitable for large service models,
- ▶ can be used for automatic composition,
- ▶ allows to generate different process languages.

Thank you for listening!  
Questions?