klase

Platform for the Composition of Web Services: Usability on an Industrial Case Study

ITC-irst
Delta Dator

Abstract. Service-Oriented Architecture (SOA) aims at integrating, in a platform independent way, functionalities spread across different organizations. Languages for specifying web service compositions, as WS-BPEL, focus on behavioural properties of processes. This opens up the possibility of applying formal methods techniques to support SOA systems. In this work we describe the application to an e-government scenario of the ASTRO framework for automated composition, verification and monitoring of web services.

Document Identifier	Deliverable 7.4
Project	MIUR-FIRB project RBNE0195K5 "Knowledge Level Automated Software Engineering"
Version	v1.0
Date	Jan 15, 2007
State	Final
Distribution	Public

Acknowledgements.

This document is part of a research project funded by the FIRB 2001 Programme of the “Ministero dell’Istruzione, dell’Università e della Ricerca” as project number RBNE0195K5.

The partners in this project are: Istituto Trentino di Cultura (Coordinator), Università degli Studi di Trento, Università degli Studi di Genova, Università degli Studi di Roma “La Sapienza”, DeltaDator S.p.A..

Executive Summary

Service-Oriented Architecture aims at integrating in a platform independent way functionalities spread across different organizations. In this context, web services are rapidly emerging as the reference paradigm for the interaction and coordination of distributed business processes. The ability to automatically compose web services, to monitor their execution, and to verify their properties is essential to achieve their practical usage. These problems can be effectively tackled using automated planning and verification techniques. As such, automated web service composition, monitoring and verification have become reference applicative scenarios for the planning and verification areas. The ASTRO toolset relies on state-of-the-art planning and verification algorithms to integrate end-to-end composition, monitoring and verification of web services specified in the standard language WS-BPEL.

Among SOA scenarios, e-Government is gaining more and more importance because of the strong need to enhance, both in terms of efficiency and transparency, the procedures of public administrations. In this deliverable we describe the application to an e-Government scenario, the payment of a municipality italian tax called TA.R.S.U., of the ASTRO framework for automated composition, verification and monitoring of web services.

Contents

Chapter 1

Introduction

The keyword "e-Government" is capturing much attention in the last years. In public administrations, there is a strong need for a deeper interoperability between different public offices/domains. Interoperability is a key factor for the efficiency and the transparency of public administration's processes with respect to citizens and companies. Web services are rapidly emerging as the reference paradigm for the interaction and coordination of distributed business processes. The ability to automatically compose web services, to verify their properties, and to monitor their execution is essential to their practical usage. On the other side, e-Government is one of the reference application areas for SOA, and offers a big potential to be exploited by distributed computing technologies such as web services.

The ASTRO toolset (http://www.astroproject.org) supports web service development by integrating state-of-the art algorithms for automated composition [PTB05, PMBT05, PTBA05], verification [KP05, KP06] and monitoring [PBB+04, BTPT06] of web services. The toolset provides end-to-end *automated composition* functionalities for web services: given a set of component services, and a composition requirement, a new "composed" service is synthesized that orchestrates the components in order to satisfy the requirement. This problem is solved by adopting planning techniques: the set of component services are converted into a planning domain, which is visited using a symbolic search algorithm. The ASTRO toolset relies on automated search techniques also for its *automated monitoring* functionality. As described in [PBB+04, BTPT06], pieces of (Java) code are synthesized to detect and signal at runtime whether the external partners behave consistently with the protocol specifications, and with user-provided properties. To perform *verification* of properties of web services, following [KP06], the ASTRO toolset relies on symbolic model checking techniques: WS-BPEL services are encoded as state transition systems, whose execution structure is exhaustively explored to discover runs that contradict a given temporal logics requirement.

The main purpose for the work presented in this deliverable is to evaluate the benefits of the ASTRO web service composition support toolset in a real-world e-Government case study. We show how, by applying ASTRO automated synthesis and monitoring

techniques, we've obtained an advantage both in terms of process efficiency (the process itself being fully automatized by handling either the "nominal" case and the principal "non-nominal" ones), and in terms of process transparency (by using a coordination agent that provides a unique and centralized interface to the choreography user/observer, the citizen). Moreover, we show that the ASTRO technology performs well with complex stateful processes as the ones typically found in e-Government scenarios.

This deliverable is structured as follows. In Ch. 2 the TA.R.S.U. case study is presented. The simplifications and adaptations over the TA.R.S.U. case is the subject of the Ch. 3.1. In Ch. 3.4 the synthesis requirements for the `TaxAgent` composite service are described. Ch. 4 presents the verification and monitoring properties defined for the TA.R.S.U. case. In Ch. 5, finally, we draw some conclusions.

Chapter 2

The Case: Citizen Tax Payment System

The payment of municipal taxes in Italy consists in a number of complex processes that involve different and independent public administration offices. In this deliverable, we will consider the process of payment of a specific municipal tax, the TA.R.S.U. (*TAssa sui Rifiuti Solidi Urbani*, that is *municipal solid waste tax*). The TA.R.S.U. is used by the municipality for the collection and the disposal of solid waste, and is calculated on the dimension of the house occupied by the citizen, and on the number of persons living in the house.

The actors involved in the TA.R.S.U. payment process are the Ufficio Tributi, the Ufficio Ragioneria and the Banca Tesoriere (see Fig. 2.1)

Ufficio Tributi calculates the tax amount, emits a payment order (*ordine di pagamento* from now on), sends a collect order (*ordine di incasso*) to the Banca Tesoriere, and waits for the reply by the Banca Tesoriere to update the citizen fiscal status (*posizione tributaria*).

The TA.R.S.U. calculation is performed batch for all citizens twice a year. The resulting file is sent to the Banca Tesoriere. The Banca Tesoriere sends back an hardcopied list containing only the insolvent citizens. For each element of this list, the Ufficio Tributi manually modifies the proper record of its information system.

Banca Tesoriere acquires the *ordine di pagamento*, generates a RID[1] charge request and forwards the payment order request to the citizen's bank. Then it notifies the Ufficio Tributi about the tax collect request (*incasso*) and, if the RID charge request succeded, it sends also to the Ufficio Ragioneria a (temporary) tax collect notification (*incasso provvisorio*). In this case, the Banca Tesoriere also waits from the Ufficio Ragioneria for a *collect authorization* (*reversale a copertura*) and finally records it.

[1] *RID*, that stands for *Rapporti Interbancari Diretti*, is an italian interbank procedure to speedup bank transactions

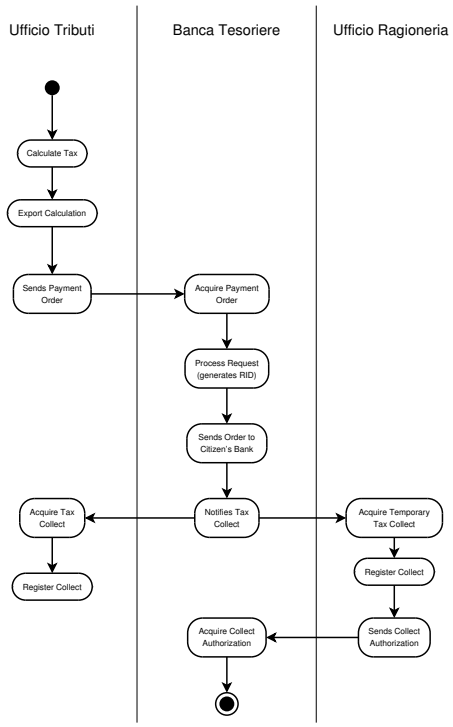


Figure 2.1: Process activity diagram.

Ufficio Ragioneria receives an *incasso provvisorio* notification, handles the *incasso* and sends a *reversale a copertura* to the Banca Tesoriere.

In the nominal case (see Fig. 2.1), the process starts with Ufficio Tributi that, given a citizen name, a time period and a type of tax, calculates the tax amount and sends a tax payment request to the Banca Tesoriere. The Banca Tesoriere handles the payment request by forwarding it to the citizen's bank, and notifies Ufficio Tributi that the tax payment request has been sent. Also, if the tax payment request has been accepted, it notifies about that the Ufficio Ragioneria that, in turn, emits and sends back to Banca Tesoriere a *reversale a copertura* (tax collect authorization).

In case of *insoluto* (the citizen has not fulfilled his tax payment request), Ufficio Tributi registers the fact that the citizen didn't pay the tax (and will eventually start other activities that don't concern the examined scenario) and terminates anyway. In this case, Ufficio Ragioneria doesn't enter the process.

It's worth noting that, in addition to these "nominal" behaviours, several "non-nominal" ones are possible. In a choreographic scenario, made up of various stateful partners, it is of utmost importance that there are no situations in which one partner aborts or hangs its execution before reaching a final state, by leaving other partners in the middle of a pending input/output activity. In real-world workflow specifications, often non-nominal cases are informally described using natural language as flat list of failures, instead of

being embodied in the protocol definition. This is what happens for the TA.R.S.U. scenario, in which a list of possible cases of failure in the backend systems calls. In order for the scenario to adhere to web services paradigm, the non-nominal cases have to be incorporated into the actors protocol. We will see in Ch. 3.1 how we've dealt this in order to automatically synthetize an agent that handles either nominal and non-nominal executions.

2.1 Implementation

Our starting situation is one where the TA.R.S.U. actors have been implemented using WS-BPEL language. WS-BPEL (*Business Process Execution Language for Web Services*) [ACD+03] is the standard language for implementing stateful web services. While WSDL allows for publishing data that are exchanged during the interactions with external processes, WS-BPEL allows for publishing the flow of the interactions with other services. A crucial property of WS-BPEL language is that components are not simply seen as atomic services, which, given some inputs, return some outputs. Instead, they are represented as stateful processes that publish an interaction protocol with external web services. Such an interaction flow is published with WS-BPEL *abstract* processes specifications. WS-BPEL also allows for the implementation of the internal process which is not published to external services, called WS-BPEL *concrete processes*. Concrete processes can be executed on standard business processes execution engines (see, e.g., [Act]). This process "web servicification" can wrap around different underlying systems by exposing in a standard way the process interface. For instance, in our context each of the three involved public administration partners (Ufficio Tributi, Ufficio Ragioneria and Banca Tesoriere) run its own software system. For example, Ufficio Tributi uses Civilia (by the KLASE partner DeltaDator), based on J2EE technology. Banca Tesoriere, instead, uses an AS/400 legacy system.

The starting point for our work of automatically composing the TA.R.S.U. services is a two-tier architecture: in the first (internal) level the functionalities provided by the three software systems are "wrapped" as sets of basic activities defined in some WSDL files. In the second level, over these *backends* services, three WS-BPEL modules are written that encapsulate the partner's functionalities as stateful processes.

2.1.1 Ufficio Tributi

The main role of an Ufficio Tributi is to manage calculation and collection of municipal taxes such as TA.R.S.U. and I.C.I. Tax amounts depend on different data such as the number of family components and the size of the house. It is Ufficio Tributi responsibility to obtain and mantain all the data needed to calculate municipal taxes amounts, and to verify that the informations declared by the citizens are true. Once the tax amounts are

computed on the available data (this happens, for example, twice a year), a tax collect request is sent to the Banca Tesoriere. Ufficio Tributi then waits for a notification of either a successfull or unsuccessfull payment and terminates (in the latter case Ufficio Tributi will eventually start or invoke a *riscossione* process that is out of the scope of this case study).

Fig. 2.2 shows, as WSDL code, the backend definition for the Ufficio Tributi service. Fig. 2.3 and Fig. 2.4 show the actual executable WS-BPEL and WSDL source code for the Ufficio Tributi service. The Ufficio Tributi backend exposes 2 syncronous operations: *CalcolaTassa* and *RegistraIncassoTassa*.

CalcolaTassa calculates a tax amount, given the citizen (*contribuente*), the time interval and the type of tax. The returned value contains the amount plus some additional information needed to automate the collect request (see `<wsdl:types>` section in Fig. 2.2)

RegistraIncassoTassa handles and records a tax payment given the tax collect code and the collect date (corresponding respectively to fields `Incasso.Rata.Codice` and `Incasso.Rata.Scadenza` of `CalcolaTassa` return message type). It returns whether or not the payment registration has been successful.

The Ufficio Tributi frontend has 2 asyncronous operations: *start* and *acquisizioneIncasso* (roughly corresponding to *CalcolaTassa* and *RegistraIncassoTassa* respectively). The control flow of the frontend service is depicted, as a state-transition system, in Fig. 2.5. The Ufficio Tributi process starts whenever it receives a tax emission request (operation `start`). The `startMsg` message type contains informations on the citizen for which the tax has to be emitted. This information is sent to the legacy backend, that extracts the citizen relevant data for tax amount calculation and performs the actual calculation. The tax amount, together with other citizen data needed to identify him, is sent for tax collection to the Banca Tesoriere. The Ufficio Tributi then waits for notification of successfull tax payment by the citizen, and then terminates.

2.1.2 Banca Tesoriere

The Banca Tesoriere service is much similar, in its functions, to a bank, and actually is often farmed out to a private bank by the municipality. The main role of the Banca Tesoriere is to care and manage the actual municipal funds. Every local public office has an account with the Banca Tesoriere. The Banca Tesoriere backend service code is shown in Fig. 2.6. The service has 5 syncronous operations:

ProcessaRichiestaIncasso receives a tax collect request, generates and sends a *RID* to the bank network, and returns a unique ID for the request.

RichiediStatoRichiestaIncasso returns a tax collect request status, given its ID.

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:s0="types.firb.deltadator.com" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tm="http://microso
ft.com/wsdl/mime/textMatching/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/w
sdl/mime/" xmlns:tns="tributi.firb.deltadator.com" xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap
.org/wsdl/soap12/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" targetNamespace="tributi.firb.deltadator.com" xmlns:wsdl="ht
tp://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="types.firb.deltadator.com">
      <s:simpleType name="CausaleIncasso">
        <s:restriction base="s:string">
          <s:enumeration value="TRIBUTI_TARSU" />
          <s:enumeration value="TRIBUTI_ICI" />
          <s:enumeration value="TRIBUTI_TOSAP" />
          <s:enumeration value="TRIBUTI_COSAP" />
          <s:enumeration value="TRIBUTI_ACQUEDOTTO" />
        </s:restriction>
      </s:simpleType>
      <s:complexType name="RichiestaDiIncasso">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Esercizio" type="s:integer" />
          <s:element minOccurs="1" maxOccurs="1" form="unqualified" name="CausaleIncasso" type="s0:CausaleIncasso" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="AnagraficaDebitore" type="s0:AnagraficaDebitore" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="AnagraficaRichiedente">
            <s:complexType>
              <s:sequence>
                <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CodiceRichiedente" type="s:string" />
              </s:sequence>
            </s:complexType>
          </s:element>
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Banca">
            <s:complexType>
              <s:sequence>
                <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="ABI" type="s:string" />
                <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CAB" type="s:string" />
                <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="ContoCorrente" type="s:string" />
                <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CIN" type="s:string" />
              </s:sequence>
            </s:complexType>
          </s:element>
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Incasso">
            <s:complexType>
              <s:sequence>
                <s:element minOccurs="1" maxOccurs="1" form="unqualified" name="ImportoTotale" type="s:decimal" />
                <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Rata" type="s0:Rata" />
              </s:sequence>
            </s:complexType>
          </s:element>
        </s:sequence>
      </s:complexType>
      <s:complexType name="AnagraficaDebitore">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CodiceUtente" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Nome" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Cognome" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Indirizzo" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CAP" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Citta" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Provincia" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CodiceFiscale" type="s:string" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="Rata">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Codice" type="s:string" />
          <s:element minOccurs="1" maxOccurs="1" form="unqualified" name="Importo" type="s:decimal" />
          <s:element minOccurs="1" maxOccurs="1" form="unqualified" name="Scadenza" type="s:date" />
        </s:sequence>
      </s:complexType>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="CalcolaTassaSoapIn">
    <wsdl:part name="codiceContribuente" type="s:string" />
    <wsdl:part name="annoInizio" type="s:date" />
    <wsdl:part name="annoFine" type="s:date" />
    <wsdl:part name="codiceRiscossione" type="s:string" />
    <wsdl:part name="tipoTassa" type="s0:CausaleIncasso" />
  </wsdl:message>
  <wsdl:message name="CalcolaTassaSoapOut">
    <wsdl:part name="infoTassa" type="s0:RichiestaDiIncasso" />
  </wsdl:message>
  <wsdl:message name="RegistraIncassoTassaSoapIn">
    <wsdl:part name="codicePagamento" type="s:string" />
    <wsdl:part name="dataPagamento" type="s:date" />
    <wsdl:part name="esitoPagamento" type="s:string" />
  </wsdl:message>
  <wsdl:message name="RegistraIncassoTassaSoapOut">
    <wsdl:part name="esito" type="s:string" />
  </wsdl:message>

  <wsdl:portType name="TributiServiceSoap">
    <wsdl:operation name="CalcolaTassa">
      <wsdl:input message="tns:CalcolaTassaSoapIn" />
      <wsdl:output message="tns:CalcolaTassaSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="RegistraIncassoTassa">
      <wsdl:input message="tns:RegistraIncassoTassaSoapIn" />
      <wsdl:output message="tns:RegistraIncassoTassaSoapOut" />
    </wsdl:operation>
  </wsdl:portType>

  <!-- Binding/deployment information omitted for readability purposes -->
</wsdl:definitions>
```

Figure 2.2: Backend WSDL for Ufficio Tributi service.

```
<?xml version="1.0" encoding="UTF-8"?>
<process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/" xmlns:abx="http://www.activebpel.org/bpel/extensio
n" xmlns:astro="urn:active-endpoints:custom_functions" xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
 xmlns:dd="types.firb.deltadator.com" xmlns:ns1="http://www.deltadator.it/BusinessProcesses/UfficioTributi" xmlns:ns2="http
://www.deltadator.it/BusinessProcesses/Tesoreria" xmlns:ns3="tributi.firb.deltadator.com" xmlns:ns5="http://schemas.xmlsoap
.org/soap/encoding/" xmlns:tns1="types.firb.deltadator.com" xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="UfficioTribu
ti" suppressJoinFailure="yes" targetNamespace="http://UfficioTributi">
  <partnerLinks>
    <partnerLink myRole="UfficioTributi_Service" name="UfficioTributi_PLT" partnerLinkType="ns1:UfficioTributi_PLT"/>
    <partnerLink name="Tesoreria_PLT" partnerLinkType="ns2:Tesoreria_PLT" partnerRole="Tesoreria_Service"/>
    <partnerLink name="TributiBackEnd_PLT" partnerLinkType="ns3:TributiBackEnd_PLT" partnerRole="TributiBackEnd"/>
  </partnerLinks>
  <variables>
    <variable messageType="ns1:startMsg" name="startMsg"/>
    <variable messageType="ns2:acquisizioneRichiestaMsg" name="acquisizioneRichiestaMsg"/>
    <variable messageType="ns1:acquisizioneIncassoMsg" name="acquisizioneIncassoMsg"/>
    <variable messageType="ns3:CalcolaTassaSoapIn" name="CalcolaTassaRequest"/>
    <variable messageType="ns3:CalcolaTassaSoapOut" name="CalcolaTassaResponse"/>
    <variable messageType="ns3:RegistraIncassoTassaSoapIn" name="RegistraIncassoTassaRequest"/>
    <variable messageType="ns3:RegistraIncassoTassaSoapOut" name="RegistraIncassoTassaResponse"/>
    <variable name="consumeActionResult" type="xsd:string"/>
  </variables>
  <correlationSets>
    <correlationSet name="CS_Tesoreria" properties="ns1:key"/>
    <correlationSet name="CS_User" properties="ns1:key"/>
  </correlationSets>
  <sequence>
    <receive name="start-CALL" operation="start" partnerLink="UfficioTributi_PLT" portType="ns1:UfficioTributi_PT" variable
="startMsg"> <correlations><correlation initiate="yes" set="CS_User"/></correlations></receive>
    <sequence>
      <assign name="Assign_HIDDEN">
        <copy><from part="codiceContribuente" variable="startMsg"/>
        <to part="codiceContribuente" variable="CalcolaTassaRequest"/></copy>
        <copy><from part="annoInizio" variable="startMsg"/>
        <to part="annoInizio" variable="CalcolaTassaRequest"/></copy>
        <copy><from part="annoFine" variable="startMsg"/>
        <to part="annoFine" variable="CalcolaTassaRequest"/></copy>
        <copy><from part="codiceRiscossione" variable="startMsg"/>
        <to part="codiceRiscossione" variable="CalcolaTassaRequest"/></copy>
        <copy><from part="tipoTassa" variable="startMsg"/>
        <to part="tipoTassa" variable="CalcolaTassaRequest"/></copy>
      </assign>
      <invoke inputVariable="CalcolaTassaRequest" name="calcolaTassa-backendcall" operation="CalcolaTassa" outputVariable="
CalcolaTassaResponse" partnerLink="TributiBackEnd_PLT" portType="ns3:TributiServiceSoap"/>
    </sequence>
    <assign name="assign_HIDDEN">
      <copy><from part="key" variable="startMsg"/>
      <to part="key" variable="acquisizioneRichiestaMsg"/></copy>
      <copy><from part="infoTassa" query="/infoTassa/Esercizio" variable="CalcolaTassaResponse"/>
      <to part="richiestaIncasso" query="/richiestaIncasso/Esercizio" variable="acquisizioneRichiestaMsg"/></copy>
      <copy><from part="infoTassa" query="/infoTassa/CausaleIncasso" variable="CalcolaTassaResponse"/>
      <to part="richiestaIncasso" query="/richiestaIncasso/CausaleIncasso" variable="acquisizioneRichiestaMsg"/></copy>
      <copy><from part="infoTassa" query="/infoTassa/AnagraficaDebitore/CodiceUtente" variable="CalcolaTassaResponse"/>
      <to part="richiestaIncasso" query="/richiestaIncasso/AnagraficaDebitore/CodiceUtente" variable="acquisizioneRichiesta
Msg"/></copy>
      <!-- ... -->
      <copy><from part="infoTassa" query="/infoTassa/AnagraficaRichiedente/CodiceRichiedente" variable="CalcolaTassaRespons
e"/>
      <to part="richiestaIncasso" query="/richiestaIncasso/AnagraficaRichiedente/CodiceRichiedente" variable="acquisizioneR
ichiestaMsg"/></copy>
      <!-- ... -->
      <copy><from part="infoTassa" query="/infoTassa/Banca/ContoCorrente" variable="CalcolaTassaResponse"/>
      <to part="richiestaIncasso" query="/richiestaIncasso/Banca/ContoCorrente" variable="acquisizioneRichiestaMsg"/></copy
>
      <copy><from part="infoTassa" query="/infoTassa/Incasso/ImportoTotale" variable="CalcolaTassaResponse"/>
      <to part="richiestaIncasso" query="/richiestaIncasso/Incasso/ImportoTotale" variable="acquisizioneRichiestaMsg"/></co
py>
      <!-- ... -->
    </assign>
    <invoke inputVariable="acquisizioneRichiestaMsg" name="invioRichiestaIncasso" operation="acquisizioneRichiesta" partner
Link="Tesoreria_PLT" portType="ns2:Tesoreria_PT">
      <correlations>
        <correlation initiate="yes" pattern="out" set="CS_Tesoreria"/>
      </correlations>
    </invoke>
    <receive name="acquisizioneIncasso" operation="acquisizioneIncasso" partnerLink="UfficioTributi_PLT" portType="ns1:Uffi
cioTributi_PT" variable="acquisizioneIncassoMsg">
      <correlations>
        <correlation set="CS_Tesoreria"/>
      </correlations>
    </receive>
    <assign name="Assign_HIDDEN">
      <copy><from part="codicePagamento" variable="acquisizioneIncassoMsg"/>
        <to part="codicePagamento" variable="RegistraIncassoTassaRequest"/>
      </copy>
      <copy><from part="dataPagamento" variable="acquisizioneIncassoMsg"/>
        <to part="dataPagamento" variable="RegistraIncassoTassaRequest"/>
      </copy>
      <copy><from expression="'PAGATO'"/>
        <to part="esitoPagamento" variable="RegistraIncassoTassaRequest"/>
      </copy>
    </assign>
    <invoke inputVariable="RegistraIncassoTassaRequest" name="registraIncasso-backendcall" operation="RegistraIncassoTassa"
 outputVariable="RegistraIncassoTassaResponse" partnerLink="TributiBackEnd_PLT" portType="ns3:TributiServiceSoap"/>
    <empty name="SUCC"/>
  </sequence>
</process>
```

Figure 2.3: Concrete WS-BPEL for Ufficio Tributi service.

```
<?xml version="1.0"?>
<definitions
    xmlns:tns="http://www.deltadator.it/BusinessProcesses/UfficioTrib
uti"
    xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-proces
s/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.deltadator.it/BusinessProcesses/Uffic
ioTributi"
    xmlns:dd="types.firb.deltadator.com" name="UfficioTributi">

  <types>
    <schema targetNamespace="types.firb.deltadator.com"
            xmlns="http://www.w3.org/2001/XMLSchema">
      <import
          namespace="http://schemas.xmlsoap.org/soap/encoding/" />

      <simpleType name="CausaleIncasso">
        <restriction base="xsd:string">
          <enumeration value="TRIBUTI_TARSU" />
          <enumeration value="TRIBUTI_ICI" />

          <enumeration value="TRIBUTI_TOSAP" />
          <enumeration value="TRIBUTI_COSAP" />
          <enumeration value="TRIBUTI_ACQUEDOTTO" />
        </restriction>
      </simpleType>

      <complexType name="AnagraficaDebitore">
        <sequence>
          <element name="CodiceUtente" nillable="true"
                   type="xsd:string" />
          <element name="Nome" nillable="true"
                   type="xsd:string" />

          <element name="Cognome" nillable="true"
                   type="xsd:string" />
          <element name="Indirizzo" nillable="true"
                   type="xsd:string" />
          <element name="CAP" type="xsd:string" />
          <element name="Citta" nillable="true"
                   type="xsd:string" />
          <element name="Provincia" nillable="true"
                   type="xsd:string" />
          <element name="CodiceFiscale" nillable="true"
                   type="xsd:anyType" />
        </sequence>
      </complexType>
      <complexType name="Rata">

        <sequence>
          <element name="Codice" nillable="true"
                   type="xsd:string" />
          <element name="Importo" nillable="true"
                   type="xsd:decimal" />
          <element name="Scadenza" nillable="true"
                   type="xsd:date" />
        </sequence>
      </complexType>
      <complexType name="RichiestaDiIncasso">
        <sequence>
          <element name="Esercizio" nillable="true"
                   type="xsd:integer" />

          <element name="CausaleIncasso" nillable="true"
                   type="dd:CausaleIncasso" />
          <element name="AnagraficaDebitore" nillable="true"
                   type="dd:AnagraficaDebitore" />
          <element name="AnagraficaRichiedente"
                   nillable="true">
            <complexType>
              <sequence>
                <element name="CodiceRichiedente"
                         nillable="true" type="xsd:string" />
              </sequence>
            </complexType>
          </element>

          <element name="Banca" nillable="true">
            <complexType>
              <sequence>
                <element name="ABI" type="xsd:string" />
                <element name="CAB" type="xsd:string" />
                <element name="ContoCorrente"
                         nillable="true" type="xsd:string" />
                <element name="CIN" type="xsd:string" />
              </sequence>
            </complexType>

          </element>
          <element name="Incasso" nillable="true">
            <complexType>
              <sequence>
                <element name="ImportoTotale"
                         nillable="true" type="xsd:decimal" />
                <element maxOccurs="unbounded"
                         name="Rata" nillable="true" type="dd:Rata" /
>
              </sequence>
            </complexType>
          </element>

        </sequence>
      </complexType>
      <complexType name="DatiParametroErrato">
        <sequence>
          <element name="valoreParametro" type="xsd:string" />
          <element name="nomeParametro" type="xsd:string" />
          <element name="descrizione" type="xsd:string" />
        </sequence>
      </complexType>
      <complexType name="E_ListaParametriNonValidi">
        <sequence>
          <element maxOccurs="unbounded" name="DatiParametro"
                   nillable="true" type="dd:DatiParametroErrato" />
          <element name="MsgErrore" nillable="true"
                   type="xsd:string" />
        </sequence>
      </complexType>
      <complexType name="E_OperazioneFallita">
        <sequence>
          <element name="codiceErrore" type="xsd:string" />

          <element name="descrizione" type="xsd:string" />
        </sequence>
      </complexType>
    </schema>
  </types>

  <message name="startMsg">
    <part name="key" type="xsd:string" />
    <part name="codiceContribuente" type="xsd:string" />
    <part name="annoInizio" type="xsd:string" />
    <part name="annoFine" type="xsd:string" />
    <part name="codiceRiscossione" type="xsd:string" />
    <part name="tipoTassa" type="xsd:string" />
  </message>
  <message name="acquisizioneIncassoMsg">
    <part name="key" type="xsd:string" />

    <!--Codice univoco del pagamento-->
    <part name="codicePagamento" type="xsd:string" />

    <!--Data dell'avvenuto pagamento-->
    <part name="dataPagamento" type="xsd:date" />

    <!-- esito pagamento -->
    <part name="esitoPagamento" type="xsd:string"/>
  </message>

  <portType name="UfficioTributi_PT">
    <operation name="start">
      <input message="tns:startMsg" />
    </operation>
    <operation name="acquisizioneIncasso">
      <input message="tns:acquisizioneIncassoMsg" />
    </operation>
  </portType>

  <!-- binding definition omitted -->

  <bpws:property name="key" type="xsd:string" />
  <bpws:propertyAlias propertyName="tns:key"
                      messageType="tns:startMsg" part="key" />
  <bpws:propertyAlias propertyName="tns:key"
                      messageType="tns:acquisizioneIncassoMsg" part="
key" />

  <plnk:partnerLinkType name="UfficioTributi_PLT">
    <plnk:role name="UfficioTributi_Service">
      <plnk:portType name="tns:UfficioTributi_PT" />
    </plnk:role>
  </plnk:partnerLinkType>
</definitions>
```

Figure 2.4: WSDL definition for Ufficio Tributi service.

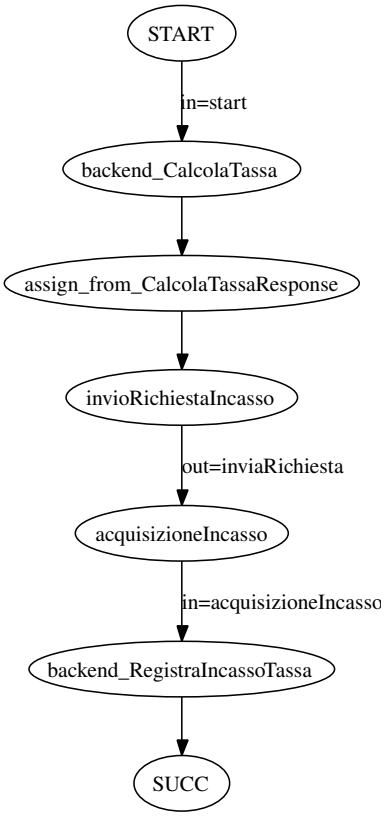


Figure 2.5: Control flow (as STS) for the Ufficio Tributi frontend service.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="ttn.firb.deltadator.com" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="ttn.firb.deltadator.com" xmlns:intf="ttn.firb.deltadator.com" xmlns:tns1="types.firb.deltadator.com" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <schema targetNamespace="types.firb.deltadator.com" xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
      <simpleType name="CausaleIncasso">
        <restriction base="xsd:string">
          <enumeration value="TRIBUTI_TARSU"/>
          <enumeration value="TRIBUTI_ICI"/>
          <enumeration value="TRIBUTI_IOSAP"/>
          <enumeration value="TRIBUTI_COSAP"/>
          <enumeration value="TRIBUTI_ACQUEDOTTO"/>
        </restriction>
      </simpleType>
      <complexType name="AnagraficaDebitore">
        <sequence>
          <element name="CodiceUtente" nillable="true" type="xsd:string"/>
          <element name="Nome" nillable="true" type="xsd:string"/>
          <element name="Cognome" nillable="true" type="xsd:string"/>
          <element name="Indirizzo" nillable="true" type="xsd:string"/>
          <element name="CAP" type="xsd:string"/>
          <element name="Citta" nillable="true" type="xsd:string"/>
          <element name="Provincia" nillable="true" type="xsd:string"/>
          <element name="CodiceFiscale" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <complexType name="Rata">
        <sequence>
          <element name="Codice" nillable="true" type="xsd:string"/>
          <element name="Importo" nillable="true" type="xsd:decimal"/>
          <element name="Scadenza" nillable="true" type="xsd:date"/>
        </sequence>
      </complexType>
      <complexType name="RichiestaDiIncasso">
        <sequence>
          <element name="Esercizio" nillable="true" type="xsd:integer"/>
          <element name="CausaleIncasso" nillable="true" type="tns1:CausaleIncasso"/>
          <element name="AnagraficaDebitore" nillable="true" type="tns1:AnagraficaDebitore"/>
          <element name="AnagraficaRichiedente" nillable="true">
            <complexType>
              <sequence>
                <element name="CodiceRichiedente" nillable="true" type="xsd:string"/>
              </sequence>
            </complexType>
          </element>
          <element name="Banca" nillable="true">
            <complexType>
              <sequence>
                <element name="ABI" type="xsd:string"/>
                <element name="CAB" type="xsd:string"/>
                <element name="ContoCorrente" nillable="true" type="xsd:string"/>
                <element name="CIN" type="xsd:string"/>
              </sequence>
            </complexType>
          </element>
          <element name="Incasso" nillable="true">
            <complexType>
              <sequence>
                <element name="ImportoTotale" nillable="true" type="xsd:decimal"/>
                <element name="Rata" nillable="true" type="tns1:Rata"/>
              </sequence>
            </complexType>
          </element>
        </sequence>
      </complexType>
      <complexType name="DatiParametroErrato">
        <sequence>
          <element name="valoreParametro" type="xsd:string"/>
          <element name="nomeParametro" type="xsd:string"/>
          <element name="descrizione" type="xsd:string"/>
        </sequence>
      </complexType>
      <complexType name="E_ListaParametriNonValidi">
        <sequence>
          <element maxOccurs="unbounded" name="DatiParametro" nillable="true" type="tns1:DatiParametroErrato"/>
          <element name="MsgErrore" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
      <complexType name="E_OperazioneFallita">
        <sequence>
          <element name="codiceErrore" type="xsd:string"/>
          <element name="descrizione" type="xsd:string"/>
        </sequence>
      </complexType>
      <simpleType name="StatoIncasso">
        <restriction base="xsd:string">
          <enumeration value="NonEsitato"/>
          <enumeration value="Pagato"/>
          <enumeration value="Insoluto"/>
          <enumeration value="Errato"/>
        </restriction>
      </simpleType>
      <simpleType name="Messaggio419">
        <restriction base="xsd:string">
          <enumeration value="TRANSAZIONE_ESEGUITA"/>
          <enumeration value="PROBLEMA"/>
          <enumeration value="ERRORE"/>
        </restriction>
      </simpleType>
      <complexType name="EsitoRichiestaIncasso">
        <sequence>
          <element maxOccurs="1" minOccurs="0" name="StatoIncasso" nillable="true" type="tns1:StatoIncasso"/>
          <element maxOccurs="1" minOccurs="0" name="DataIncasso" nillable="true" type="xsd:date"/>
          <element maxOccurs="1" minOccurs="0" name="Codice" nillable="true" type="xsd:string"/>
          <element maxOccurs="1" minOccurs="0" name="Messaggio419" nillable="true" type="tns1:Messaggio419"/>
        </sequence>
      </complexType>
      <complexType name="Incasso">
        <sequence>
          <element name="NumeroIncasso" type="xsd:int"/>
          <element name="Importo" nillable="true" type="xsd:decimal"/>
          <element name="DataIncasso" nillable="true" type="xsd:date"/>
          <element name="CausaleIncasso" nillable="true" type="tns1:CausaleIncasso"/>
        </sequence>
      </complexType>
      <complexType name="Incassi">
        <sequence>
          <element maxOccurs="unbounded" minOccurs="0" name="listaIncassi" type="tns1:Incasso"/>
          <element name="descrizione" type="xsd:string"/>
        </sequence>
      </complexType>
      <simpleType name="GestioneDatiBilancio">
        <restriction base="xsd:string">
          <enumeration value="C"/>
          <enumeration value="R"/>
        </restriction>
      </simpleType>
      <complexType name="Reversale">
        <sequence>
          <element name="CodiceEnte" nillable="true" type="xsd:string"/>
          <element name="DescrizionEnte" nillable="true" type="xsd:string"/>
          <element name="Esercizio" nillable="true" type="xsd:integer"/>
          <element name="NumeroReversale" nillable="true" type="xsd:integer"/>
          <element name="OggettoReversale" nillable="true" type="xsd:string"/>
          <element name="ImportoReversale" nillable="true" type="xsd:decimal"/>
          <element name="DataReversale" nillable="true" type="xsd:date"/>
          <element name="DatiBilancio" nillable="true">
            <complexType>
              <sequence>
                <element name="Gestione" nillable="true" type="tns1:GestioneDatiBilancio"/>
                <element name="CodiceCapitolo" nillable="true" type="xsd:string"/>
                <element name="CodiceArticolo" nillable="true" type="xsd:string"/>
                <element name="Anno" nillable="true" type="xsd:integer"/>
                <element maxOccurs="1" minOccurs="0" name="CodiceVoceEconomica" nillable="true" type="xsd:string"/>
              </sequence>
            </complexType>
          </element>
          <element maxOccurs="unbounded" name="RigaReversale" nillable="true">
            <complexType>
              <sequence>
                <element name="NumeroIncasso" nillable="true" type="xsd:integer"/>
                <element name="ImportoIncrociato" nillable="true" type="xsd:decimal"/>
                <element name="CodiceCGE" nillable="true" type="xsd:string"/>
              </sequence>
            </complexType>
          </element>
        </sequence>
      </complexType>
      <simpleType name="StatoRegistrazione">
        <restriction base="xsd:string">
          <enumeration value="InRegistrazione"/>
          <enumeration value="Registrato"/>
          <enumeration value="Fallito"/>
        </restriction>
      </simpleType>
      <complexType name="EsitoRegistrazioneReversale">
        <sequence>
          <element name="Stato" nillable="true" type="tns1:StatoRegistrazione"/>
          <element name="Descrizione" nillable="true" type="xsd:string"/>
        </sequence>
      </complexType>
    </schema>
  </wsdl:types>
  <wsdl:message name="RegistraReversaleResponse">
    <wsdl:part name="idRichiesta" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="RichiediListaIncassiResponse">
    <wsdl:part name="incassi" type="tns1:Incassi"/>
  </wsdl:message>
  <wsdl:message name="E_ListaParametriNonValidi">
    <wsdl:part name="errore" type="tns1:E_ListaParametriNonValidi"/>
  </wsdl:message>
  <wsdl:message name="E_OperazioneFallita">
    <wsdl:part name="errore" type="tns1:E_OperazioneFallita"/>
  </wsdl:message>
  <wsdl:message name="RichiediStatoRegistrazioneReversaleRequest">
    <wsdl:part name="idRichiesta" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="ProcessaRichiestaIncassoRequest">
    <wsdl:part name="richiestaIncasso" type="tns1:RichiestaDiIncasso"/>
  </wsdl:message>
  <wsdl:message name="RichiediStatoRichiestaIncassoResponse">
    <wsdl:part name="esito" type="tns1:EsitoRichiestaIncasso"/>
  </wsdl:message>
  <wsdl:message name="RichiediStatoRichiestaIncassoRequest">
    <wsdl:part name="idRichiesta" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="ProcessaRichiestaIncassoResponse">
    <wsdl:part name="idRichiesta" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="RichiediStatoRegistrazioneReversaleResponse">
    <wsdl:part name="esito" type="tns1:EsitoRegistrazioneReversale"/>
  </wsdl:message>
  <wsdl:message name="RegistraReversaleRequest">
    <wsdl:part name="reversale" type="tns1:Reversale"/>
  </wsdl:message>
  <wsdl:message name="RichiediListaIncassiRequest">
    <wsdl:part name="codiceEnte" type="xsd:string"/>
    <wsdl:part name="esercizio" type="xsd:integer"/>
    <wsdl:part name="causale" type="tns1:CausaleIncasso"/>
    <wsdl:part name="dataInizio" type="xsd:date"/>
    <wsdl:part name="dataFine" type="xsd:date"/>
  </wsdl:message>
  <wsdl:portType name="TTNPortType">
    <wsdl:operation name="ProcessaRichiestaIncasso" parameterOrder="richiestaIncasso">
      <wsdl:input message="impl:ProcessaRichiestaIncassoRequest" name="ProcessaRichiestaIncassoRequest"/>
      <wsdl:output message="impl:ProcessaRichiestaIncassoResponse" name="ProcessaRichiestaIncassoResponse"/>
      <wsdl:fault message="impl:E_ListaParametriNonValidi" name="E_ListaParametriNonValidi"/>
      <wsdl:fault message="impl:E_OperazioneFallita" name="E_OperazioneFallita"/>
    </wsdl:operation>
    <wsdl:operation name="RichiediStatoRichiestaIncasso" parameterOrder="idRichiesta">
      <wsdl:input message="impl:RichiediStatoRichiestaIncassoRequest" name="RichiediStatoRichiestaIncassoRequest"/>
      <wsdl:output message="impl:RichiediStatoRichiestaIncassoResponse" name="RichiediStatoRichiestaIncassoResponse"/>
      <wsdl:fault message="impl:E_ListaParametriNonValidi" name="E_ListaParametriNonValidi"/>
      <wsdl:fault message="impl:E_OperazioneFallita" name="E_OperazioneFallita"/>
    </wsdl:operation>
    <wsdl:operation name="RichiediListaIncassi" parameterOrder="codiceEnte esercizio causale dataInizio dataFine">
      <wsdl:input message="impl:RichiediListaIncassiRequest" name="RichiediListaIncassiRequest"/>
      <wsdl:output message="impl:RichiediListaIncassiResponse" name="RichiediListaIncassiResponse"/>
      <wsdl:fault message="impl:E_ListaParametriNonValidi" name="E_ListaParametriNonValidi"/>
      <wsdl:fault message="impl:E_OperazioneFallita" name="E_OperazioneFallita"/>
    </wsdl:operation>
    <wsdl:operation name="RegistraReversale" parameterOrder="reversale">
      <wsdl:input message="impl:RegistraReversaleRequest" name="RegistraReversaleRequest"/>
      <wsdl:output message="impl:RegistraReversaleResponse" name="RegistraReversaleResponse"/>
      <wsdl:fault message="impl:E_ListaParametriNonValidi" name="E_ListaParametriNonValidi"/>
      <wsdl:fault message="impl:E_OperazioneFallita" name="E_OperazioneFallita"/>
    </wsdl:operation>
    <wsdl:operation name="RichiediStatoRegistrazioneReversale" parameterOrder="idRichiesta">
      <wsdl:input message="impl:RichiediStatoRegistrazioneReversaleRequest" name="RichiediStatoRegistrazioneReversaleRequest"/>
      <wsdl:output message="impl:RichiediStatoRegistrazioneReversaleResponse" name="RichiediStatoRegistrazioneReversaleResponse"/>
      <wsdl:fault message="impl:E_ListaParametriNonValidi" name="E_ListaParametriNonValidi"/>
      <wsdl:fault message="impl:E_OperazioneFallita" name="E_OperazioneFallita"/>
    </wsdl:operation>
  </wsdl:portType>
  <!-- binding section omitted -->
</wsdl:definitions>
```

Figure 2.6: Backend WSDL for Banca Tesoriere service.

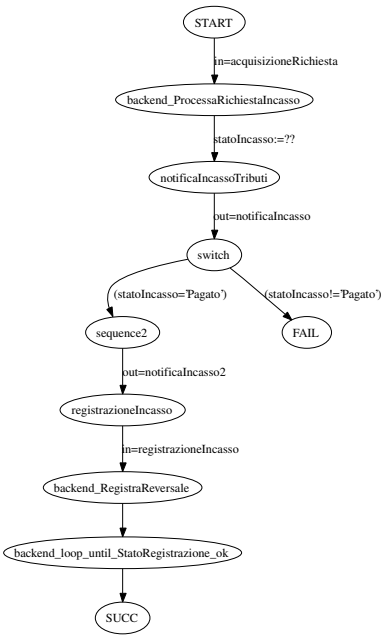


Figure 2.7: Control flow (as STS) for the Banca Tesoriere frontend service.

RichiediListaIncassi returns a list of tax collections, given a tax type, a time spam and other similar informations.

RegistraReversale records a tax payment authorization (see definition of `Reversale` complex type in Fig. 2.6) and returns an authorization request ID.

RichiediStatoRegistrazioneReversale returns, given an authorization request ID, whether the authorization has been done, is in processing or has failed.

The Banca Tesoriere frontend service flow is depicted in Fig. 2.7 (the label `statoIncasso:=??` indicates a non deterministic (opaque) variable assignment). The process starts when a payment collect request is received (operation `acquisizioneRichiesta`). The data received includes the identity of the debtor and the amount to be due to the municipality. The Banca Tesoriere forwards the collect request to the backend system that, in turn, processes the request by forwarding it to the citizen (or, in case of *RID*, by contacting the citizen's bank to obtain the payment). Banca Tesoriere then notifies the payment collect requestor that the citizen has been successfully informed about the tax collect request. If the tax payment has been made, Banca Tesoriere also notifies about that the accounting actor (Ufficio Ragioneria, in this scenario).

2.1.3 Ufficio Ragioneria

The Ufficio Ragioneria deals with municipal accounting. Its main roles are managing public budgets and handling public finances.

The Ufficio Ragioneria backend service code is shown in Fig. 2.8. The service has 2 syncronous operations:

RegistraIncassoProvvisorio receives a tax collect request, and returns a unique ID for the request.

RichiediReversaleACopertura returns, given a tax collect request ID, the corresponding tax payment authorization.

The control flow for the Ufficio Ragioneria frontend service is quite simple, and is shown in Fig. 2.9. The process receives a temporary collect notification, registers it and emits a tax payment authorization. The complete definition is given in Fig. 2.11 and in Fig. 2.10.

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:s0="types.firb.deltadator.com" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tm="http://mi
crosoft.com/wsdl/mime/textMatching/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xml
soap.org/wsdl/mime/" xmlns:tns="ragioneria.firb.deltadator.com" xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http
://schemas.xmlsoap.org/wsdl/soap12/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" targetNamespace="ragioneria.firb.delt
adator.com" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace="types.firb.deltadator.com">
      <s:simpleType name="CausaleIncasso">
        <s:restriction base="s:string">
          <s:enumeration value="TRIBUTI_TARSU" />
          <s:enumeration value="TRIBUTI_ICI" />
          <s:enumeration value="TRIBUTI_TOSAP" />
          <s:enumeration value="TRIBUTI_COSAP" />
          <s:enumeration value="TRIBUTI_ACQUEDOTTO" />
        </s:restriction>
      </s:simpleType>
      <s:complexType name="AnagraficaDebitore">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CodiceUtente" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Nome" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Cognome" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Indirizzo" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CAP" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Citta" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Provincia" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CodiceFiscale" type="s:string" />
        </s:sequence>
      </s:complexType>
      <s:complexType name="Reversale">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CodiceEnte" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="DescrizionEnte" type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Esercizio" type="s:integer" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="NumeroReversale" type="s:integer" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="OggettoReversale" type="s:string" />
          <s:element minOccurs="1" maxOccurs="1" form="unqualified" name="ImportoReversale" type="s:decimal" />
          <s:element minOccurs="1" maxOccurs="1" form="unqualified" name="DataReversale" type="s:date" />
          <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="DatiBilancio">
            <s:complexType>
              <s:sequence>
                <s:element minOccurs="1" maxOccurs="1" form="unqualified" name="Gestione" type="s0:GestioneDatiBilancio" />
                <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CodiceCapitolo" type="s:string" />
                <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CodiceArticolo" type="s:string" />
                <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="Anno" type="s:integer" />
                <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CodiceVoceEconomica" type="s:string" />
              </s:sequence>
            </s:complexType>
          </s:element>
          <s:element minOccurs="0" maxOccurs="unbounded" form="unqualified" name="RigaReversale">
            <s:complexType>
              <s:sequence>
                <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="NumeroIncasso" type="s:integer" />
                <s:element minOccurs="1" maxOccurs="1" form="unqualified" name="ImportoIncrociato" type="s:decimal" />
                <s:element minOccurs="0" maxOccurs="1" form="unqualified" name="CodiceCGE" type="s:string" />
              </s:sequence>
            </s:complexType>
          </s:element>
        </s:sequence>
      </s:complexType>
      <s:simpleType name="GestioneDatiBilancio">
        <s:restriction base="s:string">
          <s:enumeration value="C" />
          <s:enumeration value="R" />
        </s:restriction>
      </s:simpleType>
    </s:schema>
  </wsdl:types>
  <wsdl:message name="RegistraIncassoProvvisorioSoapIn">
    <wsdl:part name="importo" type="s:decimal" />
    <wsdl:part name="dataIncasso" type="s:date" />
    <wsdl:part name="origineIncasso" type="s0:CausaleIncasso" />
    <wsdl:part name="numeroProvvisorio" type="s:string" />
    <wsdl:part name="storno" type="s:boolean" />
    <wsdl:part name="esercizio" type="s:date" />
    <wsdl:part name="anagraficaVersante" type="s0:AnagraficaDebitore" />
  </wsdl:message>
  <wsdl:message name="RegistraIncassoProvvisorioSoapOut">
    <wsdl:part name="esito" type="s:string" />
  </wsdl:message>
  <wsdl:message name="RichiediReversaleACoperturaSoapIn">
    <wsdl:part name="numeroProvvisorio" type="s:string" />
  </wsdl:message>
  <wsdl:message name="RichiediReversaleACoperturaSoapOut">
    <wsdl:part name="reversale" type="s0:Reversale" />
    <wsdl:part name="disponibile" type="s:boolean" />
  </wsdl:message>
  <wsdl:portType name="RagioneriaServiceSoap">
    <wsdl:operation name="RegistraIncassoProvvisorio">
      <wsdl:input message="tns:RegistraIncassoProvvisorioSoapIn" />
      <wsdl:output message="tns:RegistraIncassoProvvisorioSoapOut" />
    </wsdl:operation>
    <wsdl:operation name="RichiediReversaleACopertura">
      <wsdl:input message="tns:RichiediReversaleACoperturaSoapIn" />
      <wsdl:output message="tns:RichiediReversaleACoperturaSoapOut" />
    </wsdl:operation>
  </wsdl:portType>
  <!-- binding section omitted -->
</wsdl:definitions>
```

Figure 2.8: Backend WSDL for Ufficio Ragioneria service.

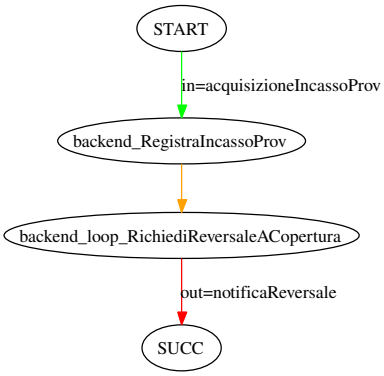


Figure 2.9: Control flow (as STS) for the Ufficio Ragioneria frontend service.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
BPEL Process Definition
Edited using ActiveWebflow(tm) Professional Designer Version 1.5.0 (http://www.active-endpoints.com)
-->
<!-- BPEL Process Definition
           -->
<!-- Edited using ActiveBPEL(tm) Designer Version 2.0.0 (http://www.active-endpoints.com) -->
<process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/" xmlns:abx="http://www.activebpel.org/bpel/extension" xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/" xmlns:dd="types.firb.deltadator.com" xmlns:nsl="http://www.deltadator.it/BusinessProcesses/UfficioRagioneria" xmlns:ns2="http://www.deltadator.it/BusinessProcesses/Tesoreria" xmlns:ns3="http://www.deltadator.it/BusinessProcesses/UfficioTributi" xmlns:ns4="ragioneria.firb.deltadator.com" xmlns:ns5="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tnsl="types.firb.deltadator.com" xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="UfficioRagioneria" suppressJoinFailure="yes" targetNamespace="http://UfficioRagioneria">
  <partnerLinks>
    <partnerLink myRole="UfficioRagioneria_Service" name="UfficioRagioneria_PLT" partnerLinkType="nsl:UfficioRagioneria_PLT"/>
    <partnerLink name="Tesoreria_PLT" partnerLinkType="ns2:Tesoreria_PLT" partnerRole="Tesoreria_Service"/>
    <partnerLink myRole="RagioneriaBackEnd" name="RagioneriaBackEnd_PLT" partnerLinkType="ns4:RagioneriaBackEnd_PLT" partnerRole="RagioneriaBackEnd"/>
  </partnerLinks>
  <variables>
    <variable messageType="nsl:acquisizioneIncassoProvMsg" name="acquisizioneIncassoProvMsg"/>
    <variable messageType="ns2:registrazioneIncassoMsg" name="registrazioneIncassoMsg"/>
    <variable messageType="ns4:RegistraIncassoProvvisorioSoapIn" name="RegistraIncassoProvvisorioRequest"/>
    <variable messageType="ns4:RegistraIncassoProvvisorioSoapOut" name="RegistraIncassoProvvisorioResponse"/>
    <variable messageType="ns4:RichiediReversaleACoperturaSoapIn" name="RichiediReversaleACoperturaRequest"/>
    <variable messageType="ns4:RichiediReversaleACoperturaSoapOut" name="RichiediReversaleACoperturaResponse"/>
    <variable name="statoReversale" type="xsd:boolean"/>
  </variables>
  <correlationSets>
    <correlationSet name="CS_UfficioRagioneria" properties="ns3:key"/>
  </correlationSets>
  <sequence>
    <receive name="acquisizioneIncassoProvvisorio" operation="acquisizioneIncassoProv" partnerLink="UfficioRagioneria_PLT" portType="nsl:UfficioRagioneria_PT" variable="acquisizioneIncassoProvMsg">
      <correlations>
        <correlation initiate="yes" set="CS_UfficioRagioneria"/>
      </correlations>
    </receive>
    <assign name="Assign_HIDDEN">
      <copy>
        <from part="importo" variable="acquisizioneIncassoProvMsg"/>
        <to part="importo" variable="RegistraIncassoProvvisorioRequest"/>
      </copy>
      <copy>
        <from part="dataIncasso" variable="acquisizioneIncassoProvMsg"/>
        <to part="dataIncasso" variable="RegistraIncassoProvvisorioRequest"/>
      </copy>
      <copy>
        <from part="origineIncasso" variable="acquisizioneIncassoProvMsg"/>
        <to part="origineIncasso" variable="RegistraIncassoProvvisorioRequest"/>
      </copy>
      <copy>
        <from part="esercizio" variable="acquisizioneIncassoProvMsg"/>
        <to part="esercizio" variable="RegistraIncassoProvvisorioRequest"/>
      </copy>
      <copy>
        <from part="anagraficaVersante" query="/anagraficaVersante/CodiceUtente" variable="acquisizioneIncassoProvMsg"/>
        <to part="anagraficaVersante" query="/anagraficaVersante/CodiceUtente" variable="RegistraIncassoProvvisorioRequest"/>
      </copy>
      <copy>
        <from part="anagraficaVersante" query="/anagraficaVersante/Nome" variable="acquisizioneIncassoProvMsg"/>
        <to part="anagraficaVersante" query="/anagraficaVersante/Nome" variable="RegistraIncassoProvvisorioRequest"/>
      </copy>
      <copy>
        <from part="anagraficaVersante" query="/anagraficaVersante/Cognome" variable="acquisizioneIncassoProvMsg"/>
        <to part="anagraficaVersante" query="/anagraficaVersante/Cognome" variable="RegistraIncassoProvvisorioRequest"/>
      </copy>
      <copy>
        <from part="anagraficaVersante" query="/anagraficaVersante/Indirizzo" variable="acquisizioneIncassoProvMsg"/>
        <to part="anagraficaVersante" query="/anagraficaVersante/Indirizzo" variable="RegistraIncassoProvvisorioRequest"/>
      </copy>
      <copy>
        <from part="anagraficaVersante" query="/anagraficaVersante/CAP" variable="acquisizioneIncassoProvMsg"/>
        <to part="anagraficaVersante" query="/anagraficaVersante/CAP" variable="RegistraIncassoProvvisorioRequest"/>
      </copy>
      <copy>
        <from part="anagraficaVersante" query="/anagraficaVersante/Citta" variable="acquisizioneIncassoProvMsg"/>
        <to part="anagraficaVersante" query="/anagraficaVersante/Citta" variable="RegistraIncassoProvvisorioRequest"/>
      </copy>
      <copy>
        <from part="anagraficaVersante" query="/anagraficaVersante/Provincia" variable="acquisizioneIncassoProvMsg"/>
        <to part="anagraficaVersante" query="/anagraficaVersante/Provincia" variable="RegistraIncassoProvvisorioRequest"/>
      </copy>
      <copy>
        <from part="anagraficaVersante" query="/anagraficaVersante/CodiceFiscale" variable="acquisizioneIncassoProvMsg"/>
        <to part="anagraficaVersante" query="/anagraficaVersante/CodiceFiscale" variable="RegistraIncassoProvvisorioRequest"/>
      </copy>
    </assign>
    <invoke inputVariable="RegistraIncassoProvvisorioRequest" name="registrazioneIncassoProvvisorio-backendcall" operation="RegistraIncassoProvvisorio" outputVariable="RegistraIncassoProvvisorioResponse" partnerLink="RagioneriaBackEnd_PLT" portType="ns4:RagioneriaServiceSoap"/>
    <assign name="Assign_HIDDEN">
      <copy>
        <from expression="false()"/>
        <to variable="statoReversale"/>
      </copy>
    </assign>
    <while condition="bpws:getVariableData('statoReversale') = false()">
      <sequence>
        <wait for="'PT1S'"/>
        <invoke inputVariable="RichiediReversaleACoperturaRequest" name="richiediReversaleCopertura-backendcall" operation="RichiediReversaleACopertura" outputVariable="RichiediReversaleACoperturaResponse" partnerLink="RagioneriaBackEnd_PLT" portType="ns4:RagioneriaServiceSoap"/>
        <assign name="Assign_HIDDEN">
          <copy>
            <from part="disponibile" variable="RichiediReversaleACoperturaResponse"/>
            <to variable="statoReversale"/>
          </copy>
        </assign>
      </sequence>
    </while>
    <assign name="assign_HIDDEN">
      <copy>
        <from part="key" variable="acquisizioneIncassoProvMsg"/>
        <to part="key" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/CodiceEnte" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/CodiceEnte" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/DescrizionEnte" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/DescrizionEnte" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/Esercizio" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/Esercizio" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/NumeroReversale" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/NumeroReversale" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/OggettoReversale" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/OggettoReversale" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/ImportoReversale" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/ImportoReversale" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/DataReversale" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/DataReversale" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/DatiBilancio/Gestione" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/DatiBilancio/Gestione" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/DatiBilancio/CodiceCapitolo" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/DatiBilancio/CodiceCapitolo" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/DatiBilancio/CodiceArticolo" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/DatiBilancio/CodiceArticolo" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/DatiBilancio/Anno" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/DatiBilancio/Anno" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/DatiBilancio/CodiceVoceEconomica" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/DatiBilancio/CodiceVoceEconomica" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/RigaReversale/NumeroIncasso" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/RigaReversale/NumeroIncasso" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/RigaReversale/ImportoIncrociato" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/RigaReversale/ImportoIncrociato" variable="registrazioneIncassoMsg"/>
      </copy>
      <copy>
        <from part="reversale" query="/reversale/RigaReversale/CodiceCGE" variable="RichiediReversaleACoperturaResponse"/>
        <to part="reversale" query="/reversale/RigaReversale/CodiceCGE" variable="registrazioneIncassoMsg"/>
      </copy>
    </assign>
    <invoke inputVariable="registrazioneIncassoMsg" name="notificaReversaleCopertura" operation="registrazioneIncasso" partnerLink="Tesoreria_PLT" portType="ns2:Tesoreria_PT">
      <correlations>
        <correlation pattern="out" set="CS_UfficioRagioneria"/>
      </correlations>
    </invoke>
    <empty name="SUCC"/>
  </sequence>
</process>
```

Figure 2.10: Concrete WS-BPEL for Ufficio Ragioneria service.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions name="UfficioRagioneria"
             targetNamespace="http://www.deltadator.it/BusinessPro
cesses/UfficioRagioneria"
             xmlns:ufficioTributi="http://www.deltadator.it/Busine
ssProcesses/UfficioTributi"
             xmlns:tns="http://www.deltadator.it/BusinessProcesses
/UfficioRagioneria"
             xmlns:dd="types.firb.deltadator.com"
             xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/bus
iness-process/"
             xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/par
tner-link/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <schema targetNamespace="types.firb.deltadator.com"
            xmlns="http://www.w3.org/2001/XMLSchema">
      <import
          namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <simpleType name="CausaleIncasso">
        <restriction base="xsd:string">
          <enumeration value="TRIBUTI_TARSU" />
          <enumeration value="TRIBUTI_ICI" />

          <enumeration value="TRIBUTI_TOSAP" />
          <enumeration value="TRIBUTI_COSAP" />
          <enumeration value="TRIBUTI_ACQUEDOTTO" />
        </restriction>
      </simpleType>
      <complexType name="AnagraficaDebitore">
        <sequence>
          <element name="CodiceUtente" nillable="true"
                   type="xsd:string" />
          <element name="Nome" nillable="true"
                   type="xsd:string" />
          <element name="Cognome" nillable="true"
                   type="xsd:string" />
          <element name="Indirizzo" nillable="true"
                   type="xsd:string" />
          <element name="CAP" type="xsd:string" />
          <element name="Citta" nillable="true"
                   type="xsd:string" />
          <element name="Provincia" nillable="true"
                   type="xsd:string" />
          <element name="CodiceFiscale" nillable="true"
                   type="xsd:anyType" />
        </sequence>
      </complexType>
      <complexType name="DatiParametroErrato">
        <sequence>
          <element name="valoreParametro" type="xsd:string" />
          <element name="nomeParametro" type="xsd:string" />
          <element name="descrizione" type="xsd:string" />
        </sequence>
      </complexType>
      <complexType name="E_ListaParametriNonValidi">
        <sequence>
          <element maxOccurs="unbounded" name="DatiParametro"
                   nillable="true" type="dd:DatiParametroErrato" /
>
          <element name="MsgErrore" nillable="true"
                   type="xsd:string" />
        </sequence>
      </complexType>
      <complexType name="E_OperazioneFallita">
        <sequence>
          <element name="codiceErrore" type="xsd:string" />
          <element name="descrizione" type="xsd:string" />
        </sequence>
      </complexType>
      <simpleType name="GestioneDatiBilancio">
        <restriction base="xsd:string">
          <enumeration value="C" />
          <enumeration value="R" />
        </restriction>
      </simpleType>
      <complexType name="Reversale">
        <sequence>
          <element name="CodiceEnte" nillable="true"
                   type="xsd:string" />
          <element name="DescrizionEnte" nillable="true"
                   type="xsd:string" />
          <element name="Esercizio" nillable="true"
                   type="xsd:integer" />
          <element name="NumeroReversale" nillable="true"
                   type="xsd:integer" />
          <element name="OggettoReversale" nillable="true"
                   type="xsd:string" />
          <element name="ImportoReversale" nillable="true"
                   type="xsd:decimal" />
          <element name="DataReversale" nillable="true"
                   type="xsd:date" />
          <element name="DatiBilancio" nillable="true">
            <complexType>
              <sequence>
                <element name="Gestione" nillable="true"
                         type="dd:GestioneDatiBilancio" />
                <element name="CodiceCapitolo"
                         nillable="true" type="xsd:string" />
                <element name="CodiceArticolo"
                         nillable="true" type="xsd:string" />
                <element name="Anno" nillable="true"
                         type="xsd:integer" />
                <element maxOccurs="1" minOccurs="0"
                         name="CodiceVoceEconomica" nillable="true
" type="xsd:string" />
              </sequence>
            </complexType>
          </element>
          <element maxOccurs="unbounded" name="RigaReversale"
                   nillable="true">
            <complexType>
              <sequence>
                <element name="NumeroIncasso"
                         nillable="true" type="xsd:integer" />
                <element name="ImportoIncrociato"
                         nillable="true" type="xsd:decimal" />
                <element name="CodiceCGE"
                         nillable="true" type="xsd:string" />
              </sequence>
            </complexType>
          </element>
        </sequence>
      </complexType>
    </schema>
  </types>
  <message name="acquisizioneIncassoProvMsg">
    <part name="key" type="xsd:string" />
    <part name="importo" type="xsd:decimal" />
    <part name="dataIncasso" type="xsd:date" />
    <part name="origineIncasso" type="dd:CausaleIncasso" />
    <part name="esercizio" type="xsd:date" />
    <part name="anagraficaVersante" type="dd:AnagraficaDebitore" /
>
  </message>
  <message name="acquisizioneIncassoMsg">
    <part name="key" type="xsd:string" />
    <part name="idIncasso" type="xsd:string" />
  </message>
  <message name="initInMsg">
    <part name="key" type="xsd:string" />
  </message>
  <message name="initOutMsg">
    <part name="pid" type="xsd:string" />
  </message>
  <portType name="UfficioRagioneria_PT">
    <operation name="init">
      <input message="tns:initInMsg" />
      <output message="tns:initOutMsg" />
    </operation>
    <operation name="acquisizioneIncassoProv">
      <input message="tns:acquisizioneIncassoProvMsg" />
    </operation>
    <operation name="acquisizioneIncasso">
      <input message="tns:acquisizioneIncassoMsg" />
    </operation>
  </portType>

  <!-- binding section omitted -->

  <plnk:partnerLinkType name="UfficioRagioneria_PLT">
    <plnk:role name="UfficioRagioneria_Service">
      <plnk:portType name="tns:UfficioRagioneria_PT" />
    </plnk:role>
  </plnk:partnerLinkType>
  <bpws:propertyAlias messageType="tns:acquisizioneIncassoMsg"
                      part="key" propertyName="ufficioTributi:key"
 />
  <bpws:propertyAlias messageType="tns:acquisizioneIncassoProvMsg"
                      part="key" propertyName="ufficioTributi:key"
 />
</definitions>
```

Figure 2.11: WSDL definition for Ufficio Ragioneria service.

Chapter 3

Automated Composition

In this Chapter we see in more detail the steps we went in to automatically syntesize the coordinator `TaxAgent` process. In Sec. 3.1 we describe how the TA.R.S.U. choreography has been reengineered in order to better suit the SOA paradigm. Sec. 3.2 motivates and presents the abstraction we've made on the actors data types. The incorporation of the non-nominal cases is the topic of the Sec. 3.3. Finally, in Sec. 3.4 we describe the TA.R.S.U. requirements definition and implementation, and we see how the executable WS-BPEL is generated automatically for `TaxAgent` out of them.

3.1 Actors Decoupling

Efficency and transparency are key motivations for e-Government ([plelT]). For what concerns efficency, in a choreographic scenario the ability to deal with non-nominal cases without human's intervention seems a crucial property for meaningful automatization. Moreover, a centralized interface to the overall process is highly suitable to gain transparency with respect to citizens. In order to obtain this, and to better fulfill the web service paradigm idea of "loosely coupled" thinking, we've reengineered the TA.R.S.U. composition architecture from a "peer-to-peer", highly coupled one to a coordinator based architecture, by introducing another entity to the choreography called `TaxAgent`, moving from the architecture depicted in Fig. 3.1 to the one depicted in Fig. 3.2.

Each process in the composition has been modified in order to contain only references to port types of its own. From an implementational point of view, this has been carried out by replacing every `invoke`/`receive` activity to other partners with a new operation with the same name.

For example, Ufficio Ragioneria invocation of the Banca Tesoriere operation `registrazioneIncasso` (see last lines of Fig. 2.10) has been replaced by an invocation of the operation `notificaReversale` that has been added to Ufficio Ragioneria WSDL (see last lines of Fig. 3.3 for the corresponding WS-BPEL code, and definition of

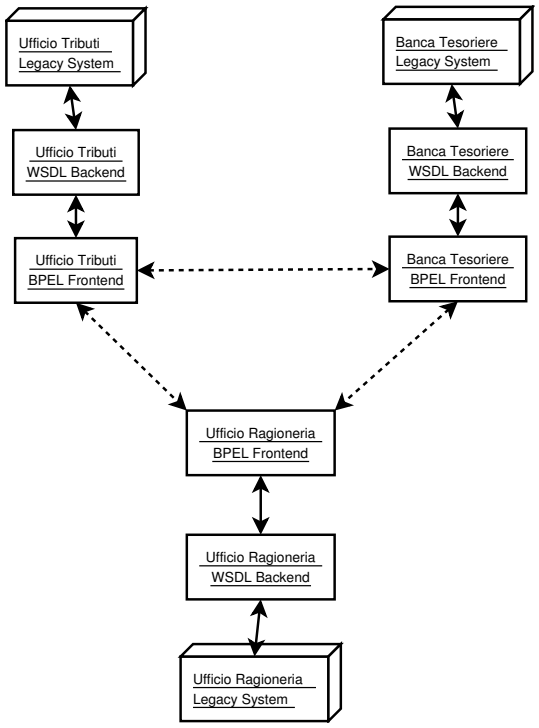


Figure 3.1: TA.R.S.U. p2p scenario.

portType `UfficioRagioneria_PT` in Fig. 3.4 for the WSDL).

The composition coordinator to be synthesized (called `TaxAgent`) is an orchestrator process whose role is to centralize the payment process, to trigger it and to notify whether the process has been successfully completed or not. The "user" of the `TaxAgent` process can actually be seen as an observer of the payment process. On completition, it is informed about the success (message `taxAck`) or failure (message `taxNack`) of the overall process. It's worth noting that, in this context, "success" means that the tax process has executed to the end in a correct and consistent way, not to the fact that the tax has been actually paid. For instance, `taxAck` will be received in case of successfull tax payment or of acknowledged *insoluto*, whether `taxNack` will be received in all other cases. The abstract WS-BPEL for the `TaxAgent` is shown as a state-transition system in Fig. 3.5.

3.2 Types Simplification

Web service composition requirements on the data flowing between the actors usually do not refer to the full information contained in the exchanged messages. As a consequence, the reasoning needed to put an orchestration in place can abstract away from data fields and structures not directly needed for specifying the "data requirements". The abstrac-

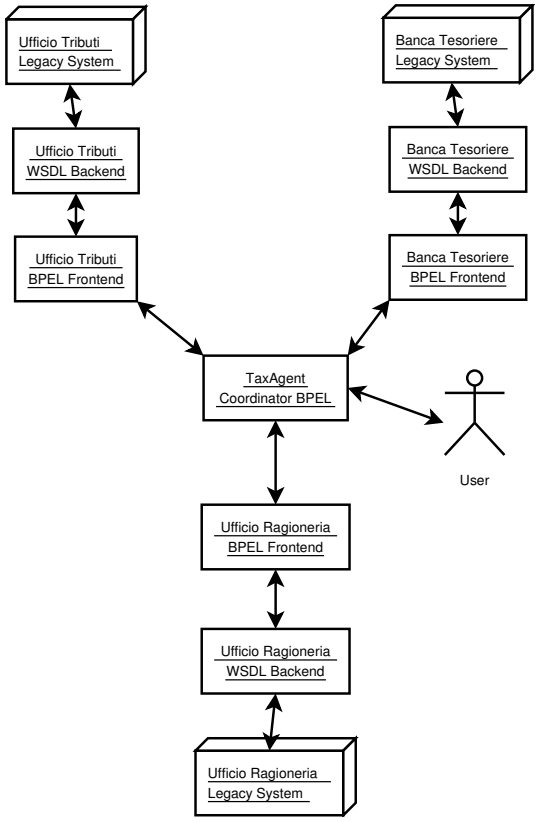


Figure 3.2: TA.R.S.U. de-coupled scenario.

```
<?xml version="1.0" encoding="UTF-8"?>
<process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
         xmlns:abx="http://www.activebpel.org/bpel/extension"
         xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
         xmlns:ns1="http://astroproject.org/BusinessProcesses/UfficioRagioneria"

         xmlns:ns5="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         abstractProcess="yes" name="UfficioRagioneria"
         suppressJoinFailure="yes" targetNamespace="http://UfficioRagioneria">
  <partnerLinks>
    <partnerLink myRole="UfficioRagioneria_Service"
                 name="UfficioRagioneria_PLT"
                 partnerLinkType="ns1:UfficioRagioneria_PLT"
                 partnerRole="UfficioRagioneria_Customer"/>
  </partnerLinks>
  <variables>
    <variable messageType="ns1:acquisizioneIncassoProvMsg"
              name="acquisizioneIncassoProvMsg"/>
    <variable messageType="ns1:notificaReversaleMsg"
              name="notificaReversaleMsg"/>
  </variables>
  <correlationSets>
    <correlationSet name="CS_User" properties="ns1:key"/>
  </correlationSets>
  <sequence>
    <receive createInstance="yes" name="acquisizioneIncassoProvvisorio"
             operation="acquisizioneIncassoProv"
             partnerLink="UfficioRagioneria_PLT"
             portType="ns1:UfficioRagioneria_PT"
             variable="acquisizioneIncassoProvMsg">
      <correlations>
        <correlation initiate="yes" set="CS_User"/>
      </correlations>
    </receive>
    <assign name="assign_HIDDEN">
      <copy>
        <from part="key" variable="acquisizioneIncassoProvMsg"/>
        <to part="key" variable="notificaReversaleMsg"/>
      </copy>
      <copy>
        <from opaque="yes"/>
        <to part="codiceReversale" variable="notificaReversaleMsg"/>
      </copy>
    </assign>
    <invoke inputVariable="notificaReversaleMsg"
            name="notificaReversaleCopertura"
            operation="notificaReversale"
            partnerLink="UfficioRagioneria_PLT"
            portType="ns1:UfficioRagioneria_CallbackPT">
      <correlations>
        <correlation pattern="out" set="CS_User"/>
      </correlations>
    </invoke>
    <empty name="SUCC"/>
  </sequence>
</process>
```

Figure 3.3: Ufficio Ragioneria Abstract WS-BPEL.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions name="UfficioRagioneria"
             targetNamespace="http://astroproject.org/BusinessProcesses/UfficioRagioneria"
             xmlns:tns="http://astroproject.org/BusinessProcesses/UfficioRagioneria"
             xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
             xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <schema targetNamespace="http://astroproject.org/BusinessProcesses/UfficioRagioneria"
            xmlns="http://www.w3.org/2001/XMLSchema">

      <element name="CausaleIncasso">
        <simpleType>
          <restriction base="xsd:string">
            <enumeration value="TRIBUTI_TARSU" />
            <enumeration value="TRIBUTI_ICI" />

            <enumeration value="TRIBUTI_TOSAP" />
            <enumeration value="TRIBUTI_COSAP" />
            <enumeration value="TRIBUTI_ACQUEDOTTO" />
          </restriction>
        </simpleType>
      </element>

    </schema>
  </types>

  <message name="acquisizioneIncassoProvMsg">
    <part name="key" type="xsd:string" />
    <part name="importo" type="xsd:string" />
    <part name="CausaleIncasso" element="tns:CausaleIncasso" />
    <part name="idVersante" type="xsd:string" />
  </message>

  <message name="notificaReversaleMsg">
    <part name="key" type="xsd:string" />
    <part name="codiceReversale" type="xsd:string" />
  </message>

  <portType name="UfficioRagioneria_PT">
    <operation name="acquisizioneIncassoProv">
      <input message="tns:acquisizioneIncassoProvMsg" />
    </operation>
  </portType>

  <portType name="UfficioRagioneria_CallbackPT">
    <operation name="notificaReversale">
      <input message="tns:notificaReversaleMsg" />
    </operation>
  </portType>

  <!-- binding section omitted -->

  <plnk:partnerLinkType name="UfficioRagioneria_PLT">
    <plnk:role name="UfficioRagioneria_Service">
      <plnk:portType name="tns:UfficioRagioneria_PT" />
    </plnk:role>
    <plnk:role name="UfficioRagioneria_Customer">
      <plnk:portType name="tns:UfficioRagioneria_CallbackPT" />
    </plnk:role>
  </plnk:partnerLinkType>


  <bpws:property name="key" type="xsd:string" />
  <bpws:propertyAlias messageType="tns:acquisizioneIncassoProvMsg"
                      part="key" propertyName="tns:key" />
  <bpws:propertyAlias messageType="tns:notificaReversaleMsg"
                      part="key" propertyName="tns:key" />

</definitions>
```

Figure 3.4: Ufficio Ragioneria new WSDL.

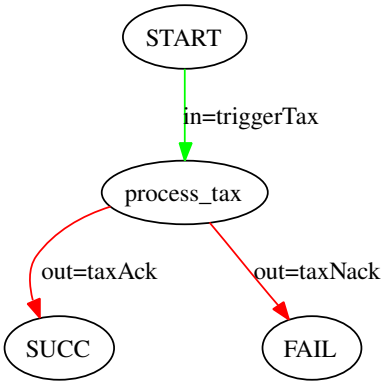


Figure 3.5: Interface of `TaxAgent` synthesized service.

tion step on data types will be performed semi-automatically (at requirement specification time) by the ASTRO toolset. For the moment, the data type abstraction is performed manually. It's worth noting that this is just a "technical" limitation, that doesn't put constraints on the structural complexity of a web service composition scenario.

ASTRO toolset supports, so far, `xsd` atomic data-types and string enumerations. Every message type in TA.R.S.U. partner definitions has been flattened and restricted to one (or, where needed, two) fields. For example (see Fig. 2.11 and Fig. 3.4), the message type `acquisizioneIncassoProvMsg` of Ufficio Ragioneria service, fields `dataIncasso`, `origineIncasso` and `esercizio` has been removed, and `anagraficaVersante` (of complex type `AnagraficaDebitore`), has been replaced by `idVersante`, of type `xsd:string`.

3.3 Non-nominal Paths

The ability to deal with non-nominal cases is a fundamental property of web service composition. However, often non-nominal cases are left implicit or only informally expressed. Moreover, the task of handling every possible combination of choreographies non-nominal behaviours is usually tedious and error-prone. One of the biggest advantages in applying automated synthesis techniques is that, once non-nominal cases are properly encoded in the actor's protocols, we have global (choreography level) non-nominal case handling almost "for free", by requiring something like: *try whatever you can to reach the composition main goal, otherwise do assure that every component fully rolls back*. In order to capture that, i.e. to allow a choreographic compensation, each actor in the composition has to expose some failure notification and cancellation functionalities in its service protocol (this can be done, for example, through dedicated operations or by using boolean fields in message types).

Within the context of the TA.R.S.U. case study, we've slightly modified the Ufficio Tributi service interface to represent the non-nominal case in which a tax payment request

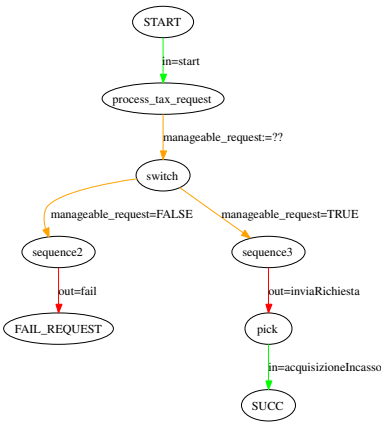


Figure 3.6: New control flow for the Ufficio Tributi service.

cannot be processed for some reason (for example, the `codiceContribuente` field on initial request does not refer to an existent citizen). If, after receiving the `start` request (see Fig. 2.3) the tax calculation with the supplied arguments is not possible, a new operation `fail` (see modified WSDL code in Fig. 3.7) is invoked and the process terminates. The modified control flow is shown in Fig. 3.6.

In order to handle the *insoluto* use case (when the tax payment request is not fulfilled by the citizen), we've added an `insoluto` operation to the Banca Tesoriere partner. The new control flow is depicted in Fig. 3.9 (to be compared with the one in Fig. 2.7).

3.4 Composition Requirements Specification

In the context of web service composition, business requirements need the ability to specify complex expected behaviors rather than just final states, and to decompose them into nominal and non-nominal ones, in order to express preferences between them. Moreover, in real life scenarios, business analysts and developers need a way to express complex requirements on data that are exchanged among component services. This is the case of our TA.R.S.U. scenario, in which we want to constrain the actors both in terms of behaviour (for example by assuring that Ufficio Tributi wont mark a tax as paid when Banca Tesoriere has determined that the citizen didn't actually pay), and in terms of data exchanged (for example by requiring that a tax account registered by Ufficio Ragioneria refers to the same citizen Ufficio Tributi has emitted the tax for).

The ASTRO toolset provides end-to-end automated composition functionalities for WS-BPEL: given a set of component web services, and a composition requirement, a new "composed" service is synthesized that orchestrates the components in order to satisfy the requirement. Within the context of ASTRO automated composition at the *knowledge level*, there is a clear separation between the specification of the data-flow requirements and that of control-flow requirements. Control-flow requirements are specified as con-

```
<?xml version="1.0"?>
<definitions
    xmlns:tns="http://astroproject.org/BusinessPro
cesses/UfficioTributi"
    xmlns:plnk="http://schemas.xmlsoap.org/ws/2003
/05/partner-link/"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:bpws="http://schemas.xmlsoap.org/ws/2003
/03/business-process/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://astroproject.org/Busin
essProcesses/UfficioTributi"
    name="UfficioTributi">

  <types>
    <schema targetNamespace="http://astroproject.o
rg/BusinessProcesses/UfficioTributi"        xmlns=
"http://www.w3.org/2001/XMLSchema">
      <element name="CausaleIncasso">
        <simpleType>
          <restriction base="xsd:string">
            <enumeration value="TRIBUTI_TARSU" />
            <enumeration value="TRIBUTI_ICI" />
            <enumeration value="TRIBUTI_TOSAP" />
            <enumeration value="TRIBUTI_COSAP" />
            <enumeration value="TRIBUTI_ACQUEDOTTO
" />
          </restriction>
        </simpleType>
      </element>

      <!-- StatoIncasso in Tesoreria.wsdl -->
      <element name="esitoPagamento">
        <simpleType>
          <restriction base="xsd:string">
            <enumeration value="NonEsitato " />
            <enumeration value="Pagato" />
            <enumeration value="Insoluto" />
            <enumeration value="Errato" />
          </restriction>
        </simpleType>
      </element>

    </schema>
  </types>

  <message name="startMsg">
    <part name="key" type="xsd:string" />
    <part name="codiceContribuente" type="xsd:stri
ng" />
    <part name="importo" type="xsd:string" />
  </message>

  <message name="acquisizioneIncassoMsg">
    <part name="key" type="xsd:string" />
    <part name="codicePagamento" type="xsd:string"
/>
    <part name="codiceDebitore" type="xsd:string"
/>
    <part name="esitoPagamento" element="tns:esito
Pagamento"/>
  </message>

  <message name="inviaRichiestaMsg">
    <part name="key" type="xsd:string" />
    <part name="CausaleIncasso" element="tns:Causa
leIncasso" />
    <part name="codiceContribuente" type="xsd:stri
ng" />
    <part name="importo" type="xsd:string" />
  </message>


  <message name="failMsg">
    <part name="key" type="xsd:string"></part>
  </message>

  <message name="fallitoIncassoMsg">
    <part name="key" type="xsd:string"></part>
  </message>

  <portType name="UfficioTributi_PT">
    <operation name="start">
      <input message="tns:startMsg" />
    </operation>
    <operation name="acquisizioneIncasso">
      <input message="tns:acquisizioneIncassoMsg"
/>
    </operation>
    <operation name="fallitoIncasso">
      <input message="tns:fallitoIncassoMsg"></inp
ut>
    </operation>
  </portType>

  <portType name="UfficioTributi_CallbackPT">
    <operation name="inviaRichiesta">
      <input message="tns:inviaRichiestaMsg" />
    </operation>
    <operation name="fail">
      <input message="tns:failMsg"></input>
    </operation>
  </portType>

  <!-- binding section omitted -->

  <bpws:propertyAlias propertyName="tns:key"
                      messageType="tns:startMsg" p
art="key" />
  <bpws:propertyAlias propertyName="tns:key"
                      messageType="tns:failMsg" pa
rt="key" />
  <bpws:propertyAlias propertyName="tns:key"
                      messageType="tns:fallitoInca
ssoMsg" part="key" />
  <bpws:propertyAlias propertyName="tns:key"
                      messageType="tns:acquisizion
eIncassoMsg" part="key" />
  <bpws:propertyAlias propertyName="tns:key"
                      messageType="tns:inviaRichie
staMsg" part="key" />
  <bpws:property name="key" type="xsd:string" />

  <plnk:partnerLinkType name="UfficioTributi_PLT">
    <plnk:role name="UfficioTributi_Service">
      <plnk:portType name="tns:UfficioTributi_PT"
/>
    </plnk:role>
    <plnk:role name="UfficioTributi_Customer">
      <plnk:portType name="tns:UfficioTributi_Call
backPT" />
    </plnk:role>
  </plnk:partnerLinkType>
  <bpws:propertyAlias messageType="tns:inviaRichie
staMsg" part="codiceContribuente" propertyName="tn
s:codiceContribuente"></bpws:propertyAlias>

</definitions>
```

Figure 3.7: Ufficio Tributi new WSDL.

```
<?xml version="1.0" encoding="UTF-8"?>
<process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
         xmlns:abx="http://www.activebpel.org/bpel/extension"
         xmlns:astro="urn:active-endpoints:custom_functions"
         xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
         xmlns:ns1="http://astroproject.org/BusinessProcesses/UfficioTributi"
         xmlns:ns5="http://schemas.xmlsoap.org/soap/encoding/"
         xmlns:tns1="types.firb.deltadator.com"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         abstractProcess="yes" name="UfficioTributi"
         suppressJoinFailure="yes" targetNamespace="http://UfficioTributi">
  <partnerLinks>
    <partnerLink myRole="UfficioTributi_Service" name="UfficioTributi_PLT" partnerLinkType="ns1:UfficioTributi_PLT" part
nerRole="UfficioTributi_Customer"/>
  </partnerLinks>
  <variables>
    <variable messageType="ns1:startMsg" name="startMsg"/>
    <variable messageType="ns1:inviaRichiestaMsg" name="inviaRichiestaMsg"/>
    <variable messageType="ns1:acquisizioneIncassoMsg" name="acquisizioneIncassoMsg"/>
    <variable name="manageable_request" type="xsd:boolean"/>
    <variable messageType="ns1:fallitoIncassoMsg" name="fallitoIncassoMsg"/>
  </variables>
  <correlationSets>
    <correlationSet name="CS_User" properties="ns1:key"/>
  </correlationSets>
  <sequence>
    <receive createInstance="yes" name="start-CALL" operation="start" partnerLink="UfficioTributi_PLT" portType="ns1:Uff
icioTributi_PT" variable="startMsg">
      <correlations>
        <correlation initiate="yes" set="CS_User"/>
      </correlations>
    </receive>
    <empty name="backend_CalcolaTassa"/>
    <assign name="CalcolaTassa_exit_code">
      <copy>
        <from opaque="yes"/>
        <to variable="manageable_request"/>
      </copy>
    </assign>
    <switch>
      <case condition="bpws:getVariableData('manageable_request') = false()">
        <sequence>
          <assign name="Prepare_failure">
            <copy>
              <from part="key" variable="startMsg"/>
              <to part="key" variable="fallitoIncassoMsg"/>
            </copy>
          </assign>
          <invoke name="notifyFailure" operation="fail" partnerLink="UfficioTributi_PLT" portType="ns1:UfficioTributi_Ca
llbackPT"/>
          <empty name="FAIL_REQUEST"/>
        </sequence>
      </case>
      <otherwise>
        <sequence>
          <assign name="assign_from_CalcolaTassaResponse">
            <copy>
              <from part="key" variable="startMsg"/>
              <to part="key" variable="inviaRichiestaMsg"/>
            </copy>
          </assign>
          <invoke inputVariable="inviaRichiestaMsg" name="invioRichiestaIncasso" operation="inviaRichiesta" partnerLink=
"UfficioTributi_PLT" portType="ns1:UfficioTributi_CallbackPT">
            <correlations>
              <correlation initiate="yes" pattern="out" set="CS_User"/>
            </correlations>
          </invoke>
          <pick>
            <onMessage operation="fallitoIncasso" partnerLink="UfficioTributi_PLT" portType="ns1:UfficioTributi_PT">
              <correlations>
                <correlation set="CS_User"/>
              </correlations>
              <empty name="FAIL"/>
            </onMessage>
            <onMessage operation="acquisizioneIncasso" partnerLink="UfficioTributi_PLT" portType="ns1:UfficioTributi_PT"
 variable="acquisizioneIncassoMsg">
              <correlations>
                <correlation set="CS_User"/>
              </correlations>
              <sequence>
                <empty name="backend_RegistraIncasso"/>
                <empty name="SUCC"/>
              </sequence>
            </onMessage>
          </pick>
        </sequence>
      </otherwise>
    </switch>
  </sequence>
</process>
```

Figure 3.8: Ufficio Tributi Abstract WS-BPEL.

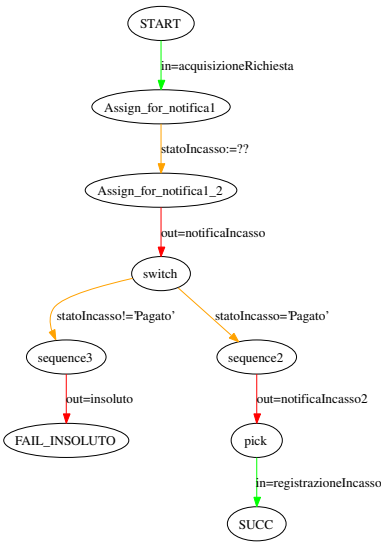


Figure 3.9: New control flow for the Banca Tesoriere service.

straints on the states of the composition partners's interface (defined by their abstract WS-BPEL). For example, we may require that, by interacting with the composition partners, the coordinator doesn't end-up leaving some of them in a "non-final" state. On the other side, requirements on the data flow are given through a set of constraints that define the valid routings and manipulations of messages that the composed service can perform. For example, we may expect that the citizen ID contained in the messages from Banca Tesoriere and Ufficio Ragioneria is the same citizen ID of the messages between Ufficio Tributi and Banca Tesoriere.

In this chapter we describe the "data" and the "flow" requirements used to synthesized the `TaxAgent` coordinator. These requirements capture 3 informal properties we want our composition to satisfy:

- the composition tries to successfully complete the tax payment process
- all services reach the end of their computation (either in a successfull or failure state), or don't start at all
- data exchanged between partners is "coherent"

3.4.1 Control-flow Requirements

As pointed out in Chapter 1, we've 2 nominal cases that can happen:

- the tax payment succeeded (with the correct amount transferred from the citizen bank to the Banca Tesoriere cash, the fiscal status of citizen set to "ok" into the

Ufficio Tributi books and the money transfer recorded into the Ufficio Ragioneria registers)

- the tax payment resulted in an *insoluto* (fiscal status of citizen set to "bad" in Ufficio Tributi books)

This requirement has been defined in terms of the partners's final states (see Fig. 3.10):

```
(UfficioTributi_pc=SUCC & UfficioRagioneria_pc = SUCC &
Tesoreria_pc = SUCC & TaxAgent_pc = SUCC) |
(UfficioTributi_pc=SUCC & UfficioRagioneria_pc = START &
Tesoreria_pc = FAIL_INSOLUTO & TaxAgent_pc = SUCC)
```

The labels SUCC, START and FAIL_INSOLUTO represent semantic annotations to the WS-BPEL code, and are encoded using standard attribute name of <empty> activity type. The string UfficioTributi_pc=SUCC represents the fact that actor Ufficio Tributi is constrained to be in state SUCC.

The "fallback" goal is, as pointed out before, to make sure that every partner finish. This is modeled with the following formula:

```
(UfficioTributi_pc=FAIL | UfficioTributi_pc=FAIL_REQUEST) &
(UfficioRagioneria_pc = START) &
(Tesoreria_pc = FAIL | Tesoreria_pc = START) &
(TaxAgent_pc = FAIL)
```

3.4.2 Data Requirements

Data flow requirements specify how output messages (messages sent to component services) are obtained from input messages (messages received from component services). This includes several important aspects: whether an input message can be used several times or just once, how several input messages must be combined to obtain an output message, whether all messages received must be processed and sent, etc..

Fig. 3.12 shows the definition of the constraints on message data for the TA.R.S.U. scenario. They are given as *data net*, i.e., as a graph where the input/output ports of the existing services are modeled as nodes, the paths in the graph define the possible routes of the messages, and the arcs define basic manipulations of these messages performed by the composed service. A graphical representation of the data requirements for TA.R.S.U. is given in Fig. 3.11. For example, the lowest element in the graph states that codiceReversale part of output operation notificaReversale of UfficioRagioneria service must be forwarded (the circled "!" represents data forwarding) to part codiceReversale of Tesoreria's input operation registrazioneIncasso. The other elements have similar meaning.

```
<domain>
  <choreography>
    <compose>
      <process_ref direction = "uses" process_id = "UfficioTributi"/>
      <process_ref direction = "uses" process_id = "UfficioRagioneria"/>
      <process_ref direction = "uses" process_id = "Tesoreria"/>
      <process_ref direction = "implements" process_id = "TaxAgent"/>
      <servicename>TaxAgent</servicename>
      <goal>
        <desc>Composition goal</desc>
        <complexgoal>
          <oneof>

            <!-- CASE tax payed -->
            <prioritygoal priority="1">
              <complexgoal>
                <simplegoal>
                  <and>
                    <simplegoal>
                      <predicate process="UfficioTributi"
                                 expr="UfficioTributi_pc = SUCC"/>
                    </simplegoal>
                    <simplegoal>
                      <predicate process="UfficioRagioneria"
                                 expr="UfficioRagioneria_pc = SUCC"/>
                    </simplegoal>
                    <simplegoal>
                      <predicate process="Tesoreria"
                                 expr="Tesoreria_pc = SUCC"/>
                    </simplegoal>
                    <simplegoal>
                      <predicate process="TaxAgent"
                                 expr="TaxAgent_pc = SUCC"/>
                    </simplegoal>
                  </and>
                </simplegoal>
              </complexgoal>
            </prioritygoal>

            <!-- CASE insoluto (nominal case, not failure) -->
            <prioritygoal priority="1">
              <complexgoal>
                <simplegoal>
                  <and>
                    <simplegoal>
                      <predicate process="UfficioTributi"
                                 expr="UfficioTributi_pc = SUCC"/>
                    </simplegoal>
                    <simplegoal>
                      <predicate process="UfficioRagioneria"
                                 expr="UfficioRagioneria_pc = START"/>
                    </simplegoal>
                    <simplegoal>
                      <predicate process="Tesoreria"
                                 expr="Tesoreria_pc = FAIL_INSOLUTO"/>
                    </simplegoal>
                    <simplegoal>
                      <predicate process="TaxAgent"
                                 expr="TaxAgent_pc = SUCC"/>
                    </simplegoal>
                  </and>
                </simplegoal>
              </complexgoal>
            </prioritygoal>


            <!-- CASE failure -->
            <prioritygoal priority="0">
              <complexgoal>
                <simplegoal>
                  <and>
                    <simplegoal>
                      <predicate process="UfficioTributi"
                                 expr="UfficioTributi_pc = FAIL | UfficioTributi_pc = FAIL_REQUEST"/>
                    </simplegoal>
                    <simplegoal>
                      <predicate process="UfficioRagioneria"
                                 expr="UfficioRagioneria_pc = START"/>
                    </simplegoal>

                    <simplegoal>
                      <predicate process="Tesoreria"
                                 expr="Tesoreria_pc = FAIL | Tesoreria_pc = START"/>
                    </simplegoal>

                    <simplegoal>
                      <predicate process="TaxAgent"
                                 expr="TaxAgent_pc = FAIL"/>
                    </simplegoal>
                  </and>
                </simplegoal>
              </complexgoal>
            </prioritygoal>

          </oneof>
        </complexgoal>

      </goal>
    </compose>
  </choreography>
</domain>
```

Figure 3.10: Control flow requirements.

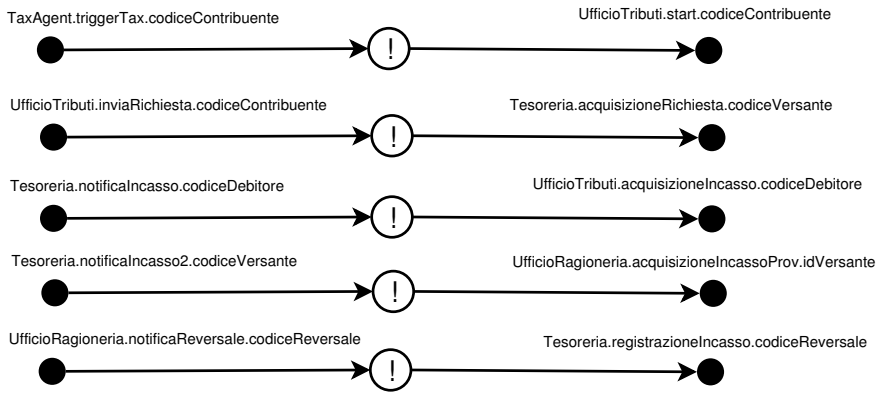


Figure 3.11: Data-net requirements.

3.4.3 Result: executable `TaxAgent`

The synthesis environment starts from the abstract WS-BPEL and WSDL of the composition partners (See Ch.3.1) and the definition of the requirements, and ends up with an executable WS-BPEL file, ready to be deployed in the WS-BPEL execution engine. The complete synthesized process is shown in Figures 3.15 and 3.16. A graphical sketch of the `TaxAgent` control-flow is depicted in Fig. 3.13.

The process starts when `TaxAgent` receives a `triggerTax` message. The request is forwarded to the Ufficio Tributi (operation `start`), that replies by sending a tax payment request (`inviaRichiesta`) to the agent. After receiving the payment request from Ufficio Tributi, the `TaxAgent` forwards it to the Banca Tesoriere, that in turn sends back a confirmation that the tax payment request has been processed (operation `notificaIncasso`), and then either a notification that the payment request has been accepted (`notificaIncasso2`) or not (operation `insoluto`). In the latter case, the Ufficio Tributi is notified of the missed tax payment, the `TaxAgent` acknowledge its user that the service interaction has been completed successfully (operation `taxAck` and the process terminates. Let's note that Ufficio Ragioneria didn't start in this use case.

In the former case (accepted payment request), the linear interaction with Ufficio Ragioneria is performed, Ufficio Tributi is sent a confirmation of the acceptance of the tax payment request, and the *reversale* (tax payment authorization) received from Ufficio Ragioneria is forwarded to Banca Tesoriere. Finally, the `TaxAgent`'s user receives an acknowledgment that the overall process is complete (`taxAck`).

If Ufficio Tributi fails to handle the initial tax request (operation `fail`), neither Banca Tesoriere nor Ufficio Ragioneria start, and `TaxAgent` terminates after sending a `taxNack` message to its user.

By comparing the `TaxAgent` abstract WS-BPEL code (Fig.3.14), handwritten as part of the TA.R.S.U. composition problem definition, with the synthesized concrete WS-BPEL (Figg.3.15 and 3.16), one can observe how the generated executable code

```
<datanet>
  <actions>
    <action id="TaxAgent_triggerTax" processId="TaxAgent"/>
    <action id="UfficioTributi_start" processId="UfficioTributi"/>
    <action id="UfficioTributi_inviaRichiesta" processId="UfficioTributi"/>
    <action id="UfficioTributi_acquisizioneIncasso" processId="UfficioTributi"/>
    <action id="Tesoreria_acquisizioneRichiesta" processId="Tesoreria"/>
    <action id="Tesoreria_notificaIncasso" processId="Tesoreria"/>
    <action id="Tesoreria_notificaIncasso2" processId="Tesoreria"/>
    <action id="Tesoreria_insoluto" processId="Tesoreria"/>
    <action id="Tesoreria_registrazioneIncasso" processId="Tesoreria"/>
    <action id="UfficioRagioneria_acquisizioneIncassoProv" processId="UfficioRagioneria"/>
    <action id="UfficioRagioneria_notificaReversale" processId="UfficioRagioneria"/>
  </actions>
  <datareqs>

    <!-- agent get req = tributi start req -->
    <datareq>
      <nodes>
        <node actionId="TaxAgent_triggerTax" id="TaxAgent_triggerTax_codiceContribuente" type="in"/>
        <node actionId="UfficioTributi_start" id="UfficioTributi_start_codiceContribuente" type="out"/>
      </nodes>
      <elements>
        <element type="identity" uid="1">
          <inputs>
            <in nodeId="TaxAgent_triggerTax_codiceContribuente"/>
          </inputs>
          <outputs>
            <out nodeId="UfficioTributi_start_codiceContribuente"/>
          </outputs>
        </element>
      </elements>
    </datareq>

    <!-- tributi send req = tesoreria get req -->
    <datareq>
      <nodes>
        <node actionId="UfficioTributi_inviaRichiesta" id="UfficioTributi_inviaRichiesta_codiceContribuente" type="in"/>
        <node actionId="Tesoreria_acquisizioneRichiesta" id="Tesoreria_acquisizioneRichiesta_codiceVersante" type="out"/>
      </nodes>
      <elements>
        <element type="filter" uid="2">
          <inputs>
            <in nodeId="UfficioTributi_inviaRichiesta_codiceContribuente"/>
          </inputs>
          <outputs>
            <out nodeId="Tesoreria_acquisizioneRichiesta_codiceVersante"/>
          </outputs>
        </element>
      </elements>
    </datareq>

    <datareq>
      <nodes>
        <node actionId="Tesoreria_notificaIncasso" id="Tesoreria_notificaIncasso_codiceDebitore" type="out"/>
        <node actionId="UfficioTributi_acquisizioneIncasso" id="UfficioTributi_acquisizioneIncasso_codiceDebitore" type="in"/>
      </nodes>
      <elements>
        <element type="filter" uid="3">
          <inputs>
            <in nodeId="Tesoreria_notificaIncasso_codiceDebitore"/>
          </inputs>
          <outputs>
            <out nodeId="UfficioTributi_acquisizioneIncasso"/>
          </outputs>
        </element>
      </elements>
    </datareq>


    <!-- tesoreria send incasso = ragioneria get incasso -->
    <datareq>
      <nodes>
        <node actionId="Tesoreria_notificaIncasso2" id="Tesoreria_notificaIncasso2_codiceVersante" type="in"/>
        <node actionId="UfficioRagioneria_acquisizioneIncassoProv" id="UfficioRagioneria_acquisizioneIncassoProv_idVersante" type="out"/>
      </nodes>
      <elements>
        <element type="filter" uid="4">
          <inputs>
            <in nodeId="Tesoreria_notificaIncasso2_codiceVersante"/>
          </inputs>
          <outputs>
            <out nodeId="UfficioRagioneria_acquisizioneIncassoProv_idVersante"/>
          </outputs>
        </element>
      </elements>
    </datareq>

    <!-- ragioneria send reversale = tesoreria get incasso -->
    <datareq>
      <nodes>
        <node actionId="UfficioRagioneria_notificaReversale" id="UfficioRagioneria_notificaReversale_codiceReversale" type="in"/>
        <node actionId="Tesoreria_registrazioneIncasso" id="Tesoreria_registrazioneIncasso_codiceReversale" type="out"/>
      </nodes>
      <elements>
        <element type="identity" uid="5">
          <inputs>
            <in nodeId="UfficioRagioneria_notificaReversale_codiceReversale"/>
          </inputs>
          <outputs>
            <out nodeId="Tesoreria_registrazioneIncasso_codiceReversale"/>
          </outputs>
        </element>
      </elements>
    </datareq>

  </datareqs>
</datanet>
```

Figure 3.12: Data flow requirements.

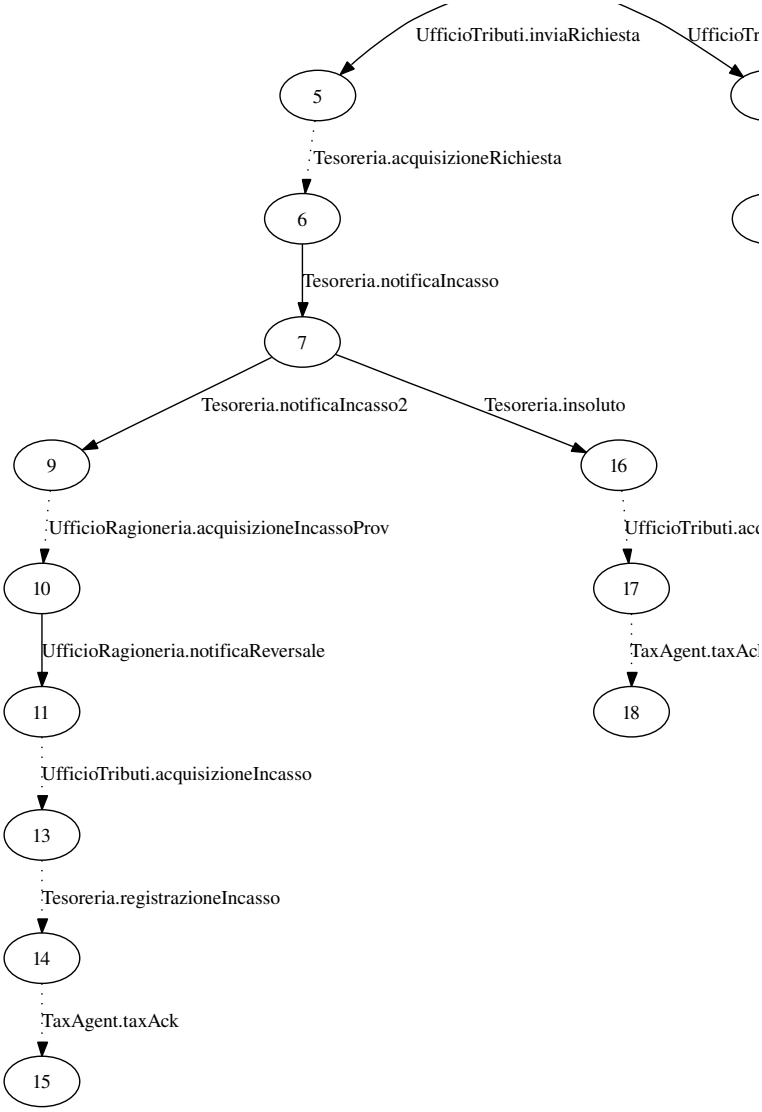


Figure 3.13: `TaxAgent` control-flow.

```
<?xml version="1.0" encoding="UTF-8"?>
<process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
         xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
         xmlns:tns="http://astroproject.org/BusinessProcesses/TaxAgent"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema"
         abstractProcess="yes" name="TaxAgent"
         suppressJoinFailure="yes" targetNamespace="http://TaxAgent">
  <partnerLinks>
    <partnerLink myRole="TaxAgent_Service" name="TaxAgent_PLT"
                 partnerLinkType="tns:TaxAgent_PLT"
                 partnerRole="TaxAgent_Customer"/>
  </partnerLinks>
  <variables>
    <variable messageType="tns:triggerTaxMsg" name="triggerTaxMsg"/>
    <variable messageType="tns:taxAckMsg" name="taxAckMsg"/>
    <variable messageType="tns:taxNackMsg" name="taxNackMsg"/>
    <variable name="process_successfull" type="xsd:boolean"/>
  </variables>
  <correlationSets>
    <correlationSet name="CS1" properties="tns:key"/>
  </correlationSets>
  <sequence>
    <receive name="get_tax_request" operation="triggerTax"
             partnerLink="TaxAgent_PLT" portType="tns:TaxAgent_PT"
             variable="triggerTaxMsg">
      <correlations>
        <correlation initiate="yes" set="CS1"/>
      </correlations>
    </receive>
    <switch>
      <case condition="process_successfull">
        <sequence>
          <invoke inputVariable="taxAckMsg" name="Acknoledge_success"
                  operation="taxAck" partnerLink="TaxAgent_PLT"
                  portType="tns:TaxAgent_CallbackPT"/>
          <empty name="SUCC"/>
        </sequence>
      </case>
      <otherwise>
        <sequence>
          <invoke inputVariable="taxNackMsg" name="Notify_failure"
                  operation="taxNack" partnerLink="TaxAgent_PLT"
                  portType="tns:TaxAgent_CallbackPT"/>
          <empty name="FAIL"/>
        </sequence>
      </otherwise>
    </switch>
  </sequence>
</process>
```

Figure 3.14: TaxAgent abstract WS-BPEL.

```
<process name="TaxAgent" targetNamespace="http://astroproject.org/BusinessProcesses/TaxAgent" suppressJoinFail
ure="yes"
         xmlns:UfficioRagioneria="http://astroproject.org/BusinessProcesses/UfficioRagioneria"
         xmlns:Tesoreria="http://astroproject.org/BusinessProcesses/Tesoreria"
         xmlns:UfficioTributi="http://astroproject.org/BusinessProcesses/UfficioTributi"
         xmlns:tns="http://astroproject.org/BusinessProcesses/TaxAgent"
         xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
         xmlns:bpelx="http://schemas.collaxa.com/bpel/extension"
         xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <partnerLinks>
    <partnerLink name="UfficioRagioneria_PLT" partnerLinkType="UfficioRagioneria:UfficioRagioneria_PLT" partne
rRole="UfficioRagioneria_Service" myRole="UfficioRagioneria_Customer"/>
    <partnerLink name="Tesoreria_PLT" partnerLinkType="Tesoreria:Tesoreria_PLT" partnerRole="Tesoreria_Service
" myRole="Tesoreria_Customer"/>
    <partnerLink name="UfficioTributi_PLT" partnerLinkType="UfficioTributi:UfficioTributi_PLT" partnerRole="Uf
ficioTributi_Service" myRole="UfficioTributi_Customer"/>
    <partnerLink name="TaxAgent_PLT" partnerLinkType="tns:TaxAgent_PLT" partnerRole="TaxAgent_Customer" myRole
="TaxAgent_Service"/>
  </partnerLinks>

  <variables>
    <variable name="_UfficioTributi_start" messageType="UfficioTributi:startMsg"/>
    <variable name="_TaxAgent_taxNack" messageType="tns:taxNackMsg"/>
    <variable name="_UfficioTributi_fail" messageType="UfficioTributi:failMsg"/>
    <variable name="_Tesoreria_acquisizioneRichiesta" messageType="Tesoreria:acquisizioneRichiestaMsg"/>
    <variable name="_Tesoreria_insoluto" messageType="Tesoreria:insolutoMsg"/>
    <variable name="_UfficioRagioneria_acquisizioneIncassoProv" messageType="UfficioRagioneria:acquisizioneInc
assoProvMsg"/>
    <variable name="_UfficioTributi_acquisizioneIncasso" messageType="UfficioTributi:acquisizioneIncassoMsg"/>
    <variable name="_Tesoreria_registrazioneIncasso" messageType="Tesoreria:registrazioneIncassoMsg"/>
    <variable name="_TaxAgent_taxAck" messageType="tns:taxAckMsg"/>
    <variable name="_UfficioRagioneria_notificaReversale" messageType="UfficioRagioneria:notificaReversaleMsg"
/>
    <variable name="_Tesoreria_notificaIncasso2" messageType="Tesoreria:notificaIncasso2Msg"/>
    <variable name="_Tesoreria_notificaIncasso" messageType="Tesoreria:notificaIncassoMsg"/>
    <variable name="_UfficioTributi_inviaRichiesta" messageType="UfficioTributi:inviaRichiestaMsg"/>
    <variable name="_TaxAgent_triggerTax" messageType="tns:triggerTaxMsg"/>
    <variable name="key" type="xsd:string"/>
  </variables>

  <correlationSets>
    <correlationSet name="_CS_UfficioRagioneria_PLT" properties="UfficioRagioneria:key"/>
    <correlationSet name="_CS_Tesoreria_PLT" properties="Tesoreria:key"/>
    <correlationSet name="_CS_UfficioTributi_PLT" properties="UfficioTributi:key"/>
    <correlationSet name="_CS_TaxAgent_PLT" properties="tns:key"/>
  </correlationSets>

  <sequence>
    <receive partnerLink="TaxAgent_PLT" portType="tns:TaxAgent_PT" operation="triggerTax" variable="_TaxAgent_
triggerTax" createInstance="yes" name="receive_TaxAgent.triggerTax">
      <correlations><correlation set="_CS_TaxAgent_PLT" initiate="yes"/></correlations>
    </receive>
    <assign><copy><from variable="_TaxAgent_triggerTax" part="key"/><to variable="key"/></copy></assign>
    <assign>
      <copy><from variable="_TaxAgent_triggerTax" part="codiceContribuente" query="/tns:codiceContribuente"/><
to variable="_UfficioTributi_start" part="codiceContribuente" query="/UfficioTributi:codiceContribuente"/></co
py>
    </assign>
    <assign><copy><from variable="key"/><to variable="_UfficioTributi_start" part="key"/></copy></assign>
    <invoke partnerLink="UfficioTributi_PLT" portType="UfficioTributi:UfficioTributi_PT" operation="start" inp
utVariable="_UfficioTributi_start" name="invoke_UfficioTributi.start">
      <correlations><correlation initiate="yes" pattern="out" set="_CS_UfficioTributi_PLT"/></correlations>
    </invoke>
    <pick>
      <onMessage partnerLink="UfficioTributi_PLT" portType="UfficioTributi:UfficioTributi_CallbackPT" operatio
n="fail" variable="_UfficioTributi_fail">
        <correlations><correlation set="_CS_UfficioTributi_PLT"/></correlations>
        <sequence>
          <assign><copy><from variable="key"/><to variable="_TaxAgent_taxNack" part="key"/></copy></assign>
          <invoke partnerLink="TaxAgent_PLT" portType="tns:TaxAgent_CallbackPT" operation="taxNack" inputVaria
ble="_TaxAgent_taxNack" name="invoke_TaxAgent.taxNack">
            <correlations><correlation pattern="out" set="_CS_TaxAgent_PLT"/></correlations>
          </invoke>
          <empty name="terminate"/>
        </sequence>
      </onMessage>
      <onMessage partnerLink="UfficioTributi_PLT" portType="UfficioTributi:UfficioTributi_CallbackPT" operatio
n="inviaRichiesta" variable="_UfficioTributi_inviaRichiesta">
        <correlations><correlation set="_CS_UfficioTributi_PLT"/></correlations>
        <sequence>
          <assign>
            <copy><from variable="_UfficioTributi_inviaRichiesta" part="codiceContribuente" query="/UfficioTri
buti:codiceContribuente"/><to variable="_Tesoreria_acquisizioneRichiesta" part="codiceVersante" query="/Tesore
ria:codiceVersante"/></copy>
          </assign>
```

Figure 3.15: Synthesized TaxAgent WS-BPEL (part 1/2).

```
          <assign><copy><from variable="key"/><to variable="_Tesoreria_acquisizioneRichiesta" part="key"/></co
py></assign>
          <invoke partnerLink="Tesoreria_PLT" portType="Tesoreria:Tesoreria_PT" operation="acquisizioneRichies
ta" inputVariable="_Tesoreria_acquisizioneRichiesta" name="invoke_Tesoreria.acquisizioneRichiesta">
            <correlations><correlation initiate="yes" pattern="out" set="_CS_Tesoreria_PLT"/></correlations>
          </invoke>
          <receive partnerLink="Tesoreria_PLT" portType="Tesoreria:Tesoreria_CallbackPT" operation="notificaIn
casso" variable="_Tesoreria_notificaIncasso" name="receive_Tesoreria.notificaIncasso">
            <correlations><correlation set="_CS_Tesoreria_PLT"/></correlations>
          </receive>
          <pick>
            <onMessage partnerLink="Tesoreria_PLT" portType="Tesoreria:Tesoreria_CallbackPT" operation="insolu
to" variable="_Tesoreria_insoluto">
              <correlations><correlation set="_CS_Tesoreria_PLT"/></correlations>
              <sequence>
                <assign><copy><from variable="key"/><to variable="_UfficioTributi_acquisizioneIncasso" part="k
ey"/></copy></assign>
                <invoke partnerLink="UfficioTributi_PLT" portType="UfficioTributi:UfficioTributi_PT" operation
="acquisizioneIncasso" inputVariable="_UfficioTributi_acquisizioneIncasso" name="invoke_UfficioTributi.acquisi
zioneIncasso">
                  <correlations><correlation pattern="out" set="_CS_UfficioTributi_PLT"/></correlations>
                </invoke>
                <assign><copy><from variable="key"/><to variable="_TaxAgent_taxAck" part="key"/></copy></assig
n>
                <invoke partnerLink="TaxAgent_PLT" portType="tns:TaxAgent_CallbackPT" operation="taxAck" input
Variable="_TaxAgent_taxAck" name="invoke_TaxAgent.taxAck">
                  <correlations><correlation pattern="out" set="_CS_TaxAgent_PLT"/></correlations>
                </invoke>
                <empty name="terminate"/>
              </sequence>
            </onMessage>
            <onMessage partnerLink="Tesoreria_PLT" portType="Tesoreria:Tesoreria_CallbackPT" operation="notifi
caIncasso2" variable="_Tesoreria_notificaIncasso2">
              <correlations><correlation set="_CS_Tesoreria_PLT"/></correlations>
              <sequence>
                <assign>
                  <copy><from variable="_Tesoreria_notificaIncasso2" part="codiceVersante" query="/Tesoreria:c
odiceVersante"/><to variable="_UfficioRagioneria_acquisizioneIncassoProv" part="idVersante" query="/UfficioRag
ioneria:idVersante"/></copy>
                </assign>
                <assign><copy><from variable="key"/><to variable="_UfficioRagioneria_acquisizioneIncassoProv"
part="key"/></copy></assign>
                <invoke partnerLink="UfficioRagioneria_PLT" portType="UfficioRagioneria:UfficioRagioneria_PT"
operation="acquisizioneIncassoProv" inputVariable="_UfficioRagioneria_acquisizioneIncassoProv" name="invoke_Uf
ficioRagioneria.acquisizioneIncassoProv">
                  <correlations><correlation initiate="yes" pattern="out" set="_CS_UfficioRagioneria_PLT"/></c
orrelations>
                </invoke>
                <receive partnerLink="UfficioRagioneria_PLT" portType="UfficioRagioneria:UfficioRagioneria_Cal
lbackPT" operation="notificaReversale" variable="_UfficioRagioneria_notificaReversale" name="receive_UfficioRa
gioneria.notificaReversale">
                  <correlations><correlation set="_CS_UfficioRagioneria_PLT"/></correlations>
                </receive>
                <assign><copy><from variable="key"/><to variable="_UfficioTributi_acquisizioneIncasso" part="k
ey"/></copy></assign>
                <invoke partnerLink="UfficioTributi_PLT" portType="UfficioTributi:UfficioTributi_PT" operation
="acquisizioneIncasso" inputVariable="_UfficioTributi_acquisizioneIncasso" name="invoke_UfficioTributi.acquisi
zioneIncasso">
                  <correlations><correlation pattern="out" set="_CS_UfficioTributi_PLT"/></correlations>
                </invoke>
                <assign>
                  <copy><from variable="_UfficioRagioneria_notificaReversale" part="codiceReversale" query="/U
fficioRagioneria:codiceReversale"/><to variable="_Tesoreria_registrazioneIncasso" part="codiceReversale" query
="/Tesoreria:codiceReversale"/></copy>
                </assign>
                <assign><copy><from variable="key"/><to variable="_Tesoreria_registrazioneIncasso" part="key"/
></copy></assign>
                <invoke partnerLink="Tesoreria_PLT" portType="Tesoreria:Tesoreria_PT" operation="registrazione
Incasso" inputVariable="_Tesoreria_registrazioneIncasso" name="invoke_Tesoreria.registrazioneIncasso">
                  <correlations><correlation pattern="out" set="_CS_Tesoreria_PLT"/></correlations>
                </invoke>
                <assign><copy><from variable="key"/><to variable="_TaxAgent_taxAck" part="key"/></copy></assig
n>
                <invoke partnerLink="TaxAgent_PLT" portType="tns:TaxAgent_CallbackPT" operation="taxAck" input
Variable="_TaxAgent_taxAck" name="invoke_TaxAgent.taxAck">
                  <correlations><correlation pattern="out" set="_CS_TaxAgent_PLT"/></correlations>
                </invoke>
                <empty name="terminate"/>
              </sequence>
            </onMessage>
          </pick>
        </sequence>
      </onMessage>
    </pick>
  </sequence>

</process>
```

Figure 3.16: Synthesized TaxAgent WS-BPEL (part 2/2).

"fills" the abstract WS-BPEL skeleton by implementing the functionalities defined in the composition requirements. In the first lines after <sequence> in Fig.3.15 and Fig.3.14, for example, after the receipment of the triggerTax message, the synthesized TaxAgent copies the field codiceContribuente from triggerTax message variable to start Ufficio Tributi message variable (according to the first element of the data net, pictured in Fig. 3.11), and then invokes Ufficio Tributistart operation. After that TaxAgent non-deterministically waits for either a fail or an inviaRichiesta message from Ufficio Tributi. In the former case it emits a taxNack and terminates. We can see that, after the emission of the taxNack message, the predicate (UfficioTributi_pc=FAIL & UfficioRagioneria_pc = START & Tesoreria_pc = FAIL & TaxAgent_pc = FAIL)) holds. This satisfies the priority="0" subgoal of the control-flow requirements of Fig. 3.10.

Chapter 4

Supporting the process: Verification and Monitoring

Both static verification and run-time monitoring of web service compositions have strong motivations. Web services provide the basis for the development and execution of business processes that are distributed over the network and available via standard interfaces and protocols [GSB+01]. Service composition [KMW03] is one of the most promising ideas underlying web services: new functionalities can be defined and implemented by combining and interacting with pre-existing services. In particular, WS-BPEL allows focusing on a key aspect for the composition of Web services: the behavior of the services. This opens up the possibility of applying a range of formal techniques to the analysis of the composition behavior, and different approaches have been defined for addressing this problem, see e.g. [FUMK03, Nak02, NM02, FBS04, PRB04].

On the other side, even properties and requirements that are verified at design time, prior to deployment and execution, can be violated at run-time. This is especially the case of service oriented applications, which are most often developed by composing services that are made available by third parties, that are autonomously developed, and that can change without notification. Moreover, some problems can be detected only at run-time. There are indeed situations that, even if admissible at design time, must be promptly revealed when they happen, e.g., the fact that a bank refuses to transfer money to a partner on-line shop.

In terms of software engineering processes, automated web service monitoring is one of the core components that, together with automated synthesis, yields to the idea of circular software development process: whenever a change in any of the composition actors is discovered by service monitors, re-synthesis of the composition coordinator (and of the monitors themselves) is triggered, without the need of a human intervention.

Speaking of our e-Government composition scenario, all the issues above are well present, since the actors protocols are rather complex and the possible composition behaviours (either at “offline” static analisys time and at run-time) are not trivial nor easy

predictable. In the following, we define some formal properties that exploit the benefits of both static verification and run-time monitoring with respect to the TA.R.S.U. case study.

4.1 Verification requirements

To perform *verification* of properties of Web services, following [KP06], the ASTRO toolset relies on symbolic model checking techniques: WS-BPEL services are encoded as state transition systems, whose execution structure is exhaustively explored to discover runs that contradict a given temporal logics requirement.

In the context of our TA.R.S.U. scenario, we've used the verification framework to prove that:

1. the composition is deadlock free
2. there is the possibility that all partners reach a successfull state
3. that Ufficio Ragioneria doesn't start on *insoluto*.

Verification properties are defined using LTL temporal logic formulas. To verify deadlock freedom there's no need to specify any property, since it is automatically formulated by using the partners abstract WS-BPEL and WSDL s.

The following specification expresses the fact that either Ufficio Tributi, Banca Tesoriere and Ufficio Ragioneria can reach their successfull final state:

$$\begin{aligned}(\mathsf{F}(\mathsf{instate}(\mathsf{TaxAgent},\mathsf{SUCC}))) \wedge \\ (\mathsf{F}(\mathsf{instate}(\mathsf{UfficioTributi},\mathsf{SUCC}))) \wedge \\ (\mathsf{F}(\mathsf{instate}(\mathsf{UfficioRagioneria},\mathsf{SUCC}))) \wedge \\ (\mathsf{F}(\mathsf{instate}(\mathsf{Tesoreria},\mathsf{SUCC})))\end{aligned}$$

(F is the standard LTL operation *eventually*).

The last property considered can be formulated as:

$$(\mathsf{F}(\mathsf{instate}(\mathsf{Tesoreria},\mathsf{FAIL\_INSOLUTO}))) -> (\mathsf{G}(\mathsf{instate}(\mathsf{Ragioneria},\mathsf{START}))))$$

All described specifications result as true.

4.2 Monitor requirements

The ASTRO toolset provides *automated monitoring* functionality. As for synthesis, this relies on automated search techniques. As described in [PBB+04, BTPT06], pieces of

(Java) code are synthesized to detect and signal at runtime whether the external partners behave consistently with the protocol specifications, and with user-provided properties. The ASTRO monitoring functionality relies on a monitor execution framework that has been implemented as an extension of the Active BPEL execution engine. This run-time monitor framework is responsible for executing the monitors associated to a given process every time an instance of that process is executed. It is also responsible for reporting the status of these monitors to the user in a convenient way. It works by "sniffing" the input/output messages directed to the process that has to be monitored and by forwarding them to the Java monitors instances.

The toolset embeds a property language that allows expressing, other than boolean properties about an execution, also statistical properties concerning some features of executions (such as e.g. how many times an event has taken place, or the average value of a given element). Such statistical properties are particularly interesting when aggregated for sets of instances of a given service, something also accounted for in the language.

For the TA.R.S.U. scenario we've considered 3 properties to be monitored: partner's protocol compliance, average time for Ufficio Tributi to respond to tax calculation requests, and global number of *insoluti* upon all tax process instances.

4.2.1 Protocol compliance

A protocol monitors detects whether a partner is not respecting its agreed flow of interaction. For some technical failure reason, for example, it could happen that Banca Tesoriere, in case of *insoluto*, after receiving an `acquisizioneRichiesta`, sends an `insoluto` message without sending a `notificaIncasso`, or terminates without sending anything, or doesn't start at all. Such behaviours break the service published interface (via its abstract WS-BPEL), and are suddenly captured by the protocol monitor, that can take the appropriate fix action (for example, by automatically notifying the service administration via a SMS message).

Protocol monitors rely only on service's abstract WS-BPEL, without the need of any other specification. For each partner of the composition, the abstract WS-BPEL is parsed and transformed into a state-transition system, that, after a power-set construction that eliminates unobservable transitions and determinize the automa, is emitted as a Java piece of code (See Fig. 4.1).

4.2.2 Number of insoluti

Monitor properties are expressed with a temporal logic language derived from Past-LTL called *RTML* (*Run-Time Monitor Language*). An *RTML* formula that counts the number of insoluti is the following:

```
import java.util.Arraylist;
import java.lang.Integer;
public class UfficioTributi_Protocol implements IProcessInstanceMonitor, IBooleanPropertyMonitor {
    private long last_ts;
    private long cur_ts;
    UfficioTributiMonitor UfficioTributi = new UfficioTributiMonitor();
    class UfficioTributiMonitor {
        private int _state;
        boolean just_born = true;
        boolean is_final = false;
        boolean is_valid = true;
        public String getErrorNode()
        {
            if(just_born||is_valid) return "UfficioTributi no error";
            if(_state==0) return "UfficioTributi protocol violation";
            if(!is_final) return "UfficioTributi protocol violation (premature termination)";
            return "UfficioTributi unhandled error";
        }
        public void init()
        {
            _state=1;
        }
        public void handleEvent(IMonitorEvent event)
        {
            if (event instanceof ProcessCloseEvent) { is_valid=just_born||(is_valid&&is_final); }

        }
        public void evolve(BpelMsg msg)
        {
            just_born = false;
            if(_state==0) { is_valid=false; }
            else if((_state==2) && ((msg.getOperation().equals("fail")))){ _state=3; is_final=true; }
            else if((_state==4) && ((msg.getOperation().equals("fallitoIncasso")))){ _state=5; is_final=true; }
            else if((_state==4) && ((msg.getOperation().equals("acquisizioneIncasso")))){ _state=6; is_final=true;
}
            else if((_state==2) && ((msg.getOperation().equals("inviaRichiesta")))){ _state=4; is_final=false; }
            else if((_state==1) && ((msg.getOperation().equals("start")))){ _state=2; is_final=false;}
            else { _state=0; is_final=true; is_valid=false; }
            return;
        }
    }
    private boolean is_valid=true;
    public UfficioTributi_Protocol()
    {
        init();
    }
    public void init()
    {
        last_ts = System.currentTimeMillis();
        UfficioTributi.init();
    }
    public void handleEvent(IMonitorEvent event)
    {
        UfficioTributi.handleEvent(event);
        is_valid = ( UfficioTributi.is_valid);
    }
    public boolean isValid() { return is_valid; }
    public void evolve(BpelMsg msg)
    {
        boolean once=false;
        cur_ts = new Long(msg.getTimestamp()).longValue();
        if(msg.getSenderType().indexOf("UfficioTributi")==0 || msg.getReceiverType().indexOf("UfficioTributi")==0) { on
ce=true; UfficioTributi.evolve(msg);}
        if(once){
            is_valid = ( UfficioTributi.is_valid);
            last_ts = cur_ts;
        }
        return;
    }
    public String getErrorNode()
    {
        String errstr="";
        if(is_valid) return "no error";
        if(!UfficioTributi.is_valid) errstr = errstr + UfficioTributi.getErrorNode() + ".";
        return errstr;
    }
    public String getProcessName() { return "UfficioTributi"; }
    public String getProperty() { return "Protocol"; }
    public String getDescription() { return ""; }
}
```

Figure 4.1: Ufficio Tributi protocol monitor auto-generated code.

```
import org.astroproject.monitor.utility.Utility;
public class Class_Tesoreria_CountInsoluti implements IProcessClassMonitor, IStatist
icPropertyMonitor {
    // Formula: "COUNT  O MSG Tesoreria.output = insoluto"
    public void update() {
        float next[] = new float[1];
        next[0]=Utility.computeCount("monitor.instance.Tesoreria__CountInsoluti_0");
        state=next;
    }
    public void init() {
        state[0]=Utility.computeCount("monitor.instance.Tesoreria__CountInsoluti_0");
    }
    float state[] = new float[1];
    public float value() { return state[0]; }
    public String getErrorNode() { return "no error"; }
    public String getProcessName() { return "Tesoreria"; }
    public String getProperty() { return "CountInsoluti"; }
    public String getDescription() { return ""; }
}
```

Figure 4.2: `Class_Tesoreria_CountInsoluti.java`.

$$\mathsf{Count}\,(\mathsf{O}\,\mathsf{msg}(\mathsf{Tesoreria.output} = \mathsf{insoluto}))$$

$\mathsf{msg}(\mathsf{Tesoreria.output} = \mathsf{insoluto})$ identifies the fact that a message for operation `insoluto` has just been sent by Banca Tesoriere. $\mathsf{O}\,P$ is a (past) LTL construct that stands for "once P". Intuitively it means that formula P has been true (at least once) in the past. $\mathsf{Count}\,P$ is an instance aggregation construct. It counts how many times the instance property P has been true over all process instances since process deployment.

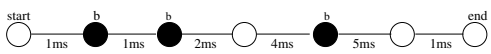
A "class" property monitor (that is a property on all service instances instead of a single instance) is translated into 2 Java sources, one for the class property and one for the underlying instance property ($\mathsf{O}\,\mathsf{msg}(\mathsf{Tesoreria.output} = \mathsf{insoluto})$, in this case). The complete Java code for `InsolutiCount` property monitor is given in Fig. 4.2 (class property) and Fig. 4.3 (corresponding instance property).

4.2.3 Tax calculation time

An *RTML* formula for computing the average time spent by Ufficio Tributi to reply to initial tax request is the following:

$$\mathsf{Avrg}\,(\mathsf{time}\,\mathsf{msg}(\mathsf{UfficioTributi.input} = \mathsf{start}))$$

The instance operator $\mathsf{time}\,P$ counts the sum of time-spans after occurrence of P, until the next relevant event for the process P is referred to. Consider for instance the sequence of events modeled in the following diagram, corresponding to the execution of a fictitious WS-BPEL process:



We have marked in black the events satisfying condition b, and we have annotated the duration of the time intervals between events. At the end of the execution we have

```
import java.util.ArrayList;
import java.lang.Integer;
public class Tesoreria__CountInsoluti_0 implements IProcessInstanceMonitor, IServant
Monitor, IBooleanPropertyMonitor {
    //  O MSG Tesoreria.output = insoluto
    class Property {
        float state[] = new float[3];
        boolean is_valid;
        void evolve(BpelMsg msg)
        {
            float next[] = new float[3];
            next[1]=((msg.getReceiverType().indexOf("Tesoreria")==0)&&msg.getOpera
tion().equals("insoluto"))?1:0;
            next[2]=((next[1]==1)||(state[2]==1))?1:0;
            is_valid=(next[2]==1);
            state=next;
            return;
        }
        void init()
        {
            state[1]=0;
            state[2]=state[1];
            is_valid=(state[2]==1);
            return;
        }
        void terminate() { return ; }

    }
    Property p = new Property();
    private boolean is_valid=true;
    public Tesoreria__CountInsoluti_0()
    {
        init();
    }
    public void init()
    {
        p.init();
    }
    public void handleEvent(IMonitorEvent event)
    {
        p.terminate();
        is_valid = p.is_valid;
    }
    public boolean isValid() { return is_valid; }
    public void evolve(BpelMsg msg)
    {
        boolean once=false;
        if(msg.getSenderType().indexOf("Tesoreria")==0 || msg.getReceiverType().in
dexOf("Tesoreria")==0) { once=true;}
        if(once){
            p.evolve(msg);
            is_valid = ((false) || p.is_valid);
        }
        return;
    }
    public String getErrorNode()
    {
        String errstr="";
        if(is_valid) return "no error";
        if(!p.is_valid) errstr = errstr + "Property CountInsoluti_0 violated. ";
        return errstr;
    }
    public String getProcessName() { return "Tesoreria_"; }
    public String getProperty() { return "CountInsoluti_0"; }
    public String getDescription() { return ""; }
}
```

Figure 4.3: `Tesoreria__CountInsoluti_0.java`.

$\mathsf{count}\,(b) = 3$ (b has been true 3 times) and $\mathsf{time}\,(b) = 8ms$ (corresponding to the sum of durations $1ms + 2ms + 5ms$).

In our case, $\mathsf{time}\,\mathsf{msg}(\mathsf{UfficioTributi.input} = \mathsf{start})$ returns the time-span between the receival of a message `start` and one of the possible next observable events for Ufficio Tributi, that are sending a `fail` or sending an `inviaRichiesta`.

$\mathsf{Avrg}\,P$ computes the average of numerical property P across all instances since process deployment.

Chapter 5

Conclusions

In this work we've applied the ASTRO toolset to an e-government scenario in order to evaluate the benefits of automated composition, verification and monitoring of web services. The studied public administration scenario was rather complex. By defining some composition requirements, both from a "data" and from a "flow" point of view, we've successfully synthesized a composition coordinator `TaxAgent` that, in addition to the nominal success case, can handle also some non nominal cases by driving all the composition partners to a final state. Another advantage of having a process coordinator is that we gain transparency, a key aspect of public administration processes: the synthesized `TaxAgent` "embeds" knowledge on the status of the interacting partners. Then we've assured, by using formal verification, that the scenario doesn't suffer of deadlocks and can finish successfully. Finally, we've defined some monitoring properties that provide, in real-time, some valuable correctness and performance indicators over the process instances.

These results enforce our belief that automated composition, verification and monitoring is a crucial aspect of service-oriented computing, and motivate the effort to continue the work on the ASTRO toolset, either on extending the WS-BPEL features that can be handled by the toolset, and on expanding the expressiveness of its languages for defining composition, verification and monitor requirements.

Bibliography

[ACD+03] T. Andrews, F. Curbera, H. Dolakia, J. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weeravarana. Business Process Execution Language for Web Services (version 1.1), 2003.

[Act] ActiveBPEL. The Open Source BPEL Engine - http://www.activebpel.org.

[BTPT06] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti. Run-Time Monitoring of Instances and Classes of Web Service Compositions. In *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2006.

[FBS04] X. Fu, T. Bultan, and J. Su. Analysis of Interacting BPEL Web Services. In *Proc. WWW'04*, 2004.

[FUMK03] H. Foster, S. Uchitel, J. Magee, and J. Kramer. Model-based verification of Web Service Compositions. In *Proc. ASE'03*, 2003.

[GSB+01] S. Graham, S. Simenov, T. Boubez, G. Daniels, D. Davis, Y. Nakamura, and R. Neyama. *Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI*. Sams, 2001.

[KMW03] R. Khalaf, N. Mukhi, and S. Weerawarana. Service Oriented Composition in BPEL4WS. In *Proc. WWW2003*, 2003.

[KP05] Raman Kazhamiakin and Marco Pistore. A Parametric Communication Model for the Verification of BPEL4WS Compositions. In *Proc. WS-FM'05*, 2005.

[KP06] "Raman Kazhamiakin, Marco Pistore, and Luca Santuari ". "analysis of communication models in web service compositions". In *"Proc. of WWW'06"*, 2006.

[Nak02] S. Nakajima. Model-checking verification for reliable web service. In *Proc. OOPSLA'02 Workshop on OOWS*, 2002.

[NM02] S. Narayanan and S. McIlraith. Simulation, Verification and Automated Composition of Web Services. In *Proc. WWW'02*, 2002.

[PBB+04] M. Pistore, P. Bertoli, F. Barbon, D. Shaparau, and P. Traverso. Planning and Monitoring Web Service Composition. In *Proc. AIMSA'04*, 2004.

[plelT] Ministro per l'Innovazione e le Tecnologie. DPCM 14 Febbrario 2002.

[PMBT05] M. Pistore, A. Marconi, P. Bertoli, and P. Traverso. Automated Composition of Web Services by Planning at the Knowledge Level. In *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2005.

[PRB04] M. Pistore, M. Roveri, and P. Busetta. Requirements-Driven Verification of Web Services. In *Proc. WS-FM'04, ENTCS*, 2004.

[PTB05] M. Pistore, P. Traverso, and P. Bertoli. Automated Composition of Web Services by Planning in Asynchronous Domains. In *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, 2005.

[PTBA05] M. Pistore, P. Traverso, P. Bertoli, and A.Marconi. Automated Synthesis of Composite BPEL4WS Web Services. In *Proc. ICWS'05*, 2005.