U.P.B. Sci. Bull., Series C, Vol. 77, Iss. 2, 2015 ISSN 2286-3540

CHANGE IMPACT ANALYSIS IN WS-BPEL PROCESSES

Cristian DĂNILĂ[1], Aurelian Mihai STĂNESCU[2]

In critical situations in which immediate decisions must be made there is a need for a tool able to analyze the impact of changes. The logic and data flow in WS-BPEL processes is quite difficult to deduce without a rigorous description. We have developed a change impact analysis tool that identifies the activities affected by the operations of deletion/addition/replacement of activities and preconditions for the addition activities. Also, if the communication activities invoke other BPEL processes, these processes are identified by name and displayed.

Keywords: WS-BPEL, change impact analysis, SOA

1. Introduction

Future Internet Enterprise Systems (FInES) describes "a field of activity with the aim to enabling enterprises, including SMEs, by means of ICT, to exploit the full potential of the Future Internet" [1]. The development of Future Internet is based on four pillars: Internet of Services (IoS), Internet of Things (IoT), Internet of Knowledge (IoK) and Internet by / for People (IoP). Everything is available as a service in the IoS research domain, software as a service becoming increasingly significant in the last years due to increased usage of SOA [2].

Service-based applications (SBA) are built on SOA architecture. A Web service has an interface described in a machine processable format (specifically WSDL [3]). Other systems interact with the web service using HTTP or XML format serialized data in conjunction with other Web standards. Most common standards used by web services are WSDL, SOAP [4], WS-BPEL [5]. An important property of SOA web services is their composability [6]. Four models of service composition can be identified [7]: service orchestration, choreography of services, coordination of services, services assembly.

In this paper the effort is focused on the orchestration of services and WS-BPEL language. WS-BPEL is the standard language for modeling service orchestrations behavior. WS-BPEL language consists of basic activities: invoke, receive, reply, assign, throw, wait, empty and structured activities: sequence, if, while, repeatUntill, pick, flow, foreach. Basic activities are atomic and describe

[1] Eng., Department of Automatic Control and Industrial Informatics University POLITEHNICA of Bucharest, România, e-mail:danila.cristian.84@gmail.com

[2] Prof., Department of Automatic Control and Industrial Informatics University POLITEHNICA of Bucharest, România, e-mail: amstanescu@yahoo.com

the basic steps of the process behavior. Structured activities encode control-flow logic. They may also contain basic activities and structured activities (e.g. more activities of "sequence" type in a "flow").

A study was performed on existing tools on the market that allow the design of WS-BPEL processes. The point of interest was represented by the capability of the analyzed tools to identify the impact of changes in the WS-BPEL process. The most popular instrument is the Eclipse BPEL Designer. It is an open-source software. Two major software manufacturers offer fee tools for designing BPEL processes: Oracle and IBM. The product marketed by Oracle is Oracle BPEL Process Manager and the product marketed by IBM is IBM Integration Designer. Modelio BPEL Designer [8] is a graphical design tool that allows the design of processes according to the BPMN standard and the generation of BPEL code based on them. Eclipse BPEL Designer provides support for the design of WS-BPEL processes [9]. The tool does not support the impact analysis that a certain change in the process may have on the concerned process activities or on other processes. The latest version available for the Oracle BPEL Process Manager is 11.1.1.7.0 [10]. This version could not be downloaded from the Oracle download site. Only customers that have support contracts with Oracle can download this version. Version 10.1.3.3 was downloaded and analyzed. Oracle BPEL Process Manager 10.1.3.3 comes pre-installed as a plugin for JDeveloper 10.1.3.3. Based on the study made, we have seen that Oracle BPEL Process Manager does not support change impact analysis in the process. The difference between the analyzed version (10.1.3.3) and the latest version, as Oracle states [11], [12] is represented by the introduction of conditional correlations aggregation scenarios. For IBM, a trial version of IBM Integration Designer tool 7.5.1 could not be downloaded [13]; it is only available for companies. According to the documentation provided by IBM [14], IBM Integration Designer tool does not assess the impact that a certain change in the process may have on the process activities concerned or on other processes. Changes made from one version to another version of the BPEL process model can be exported and imported in IBM Integration Designer Business Modeler. In the IBM Business Modeler tool changes can be analyzed, viewing the differences between process model created in Integration Designer and actual implementation. These changes can be propagated very easily to the implementation or rejected [15]. In Modelio the impact of changes in a process is not considered.

In service-based systems, there may be complex dependencies between services and business processes. Change management is a difficult problem in the SOA due to the dynamic nature of the services and their distribution [16]. Web services are coupled with business processes that contain them. The proposed approach is based on models of service-oriented business processes that capture the dependency between services and business processes. A number of patterns of change impact are identified based on changes in the process. Propagation of changes in WS-BPEL business processes can be identified by the proposed patterns. Based on the taxonomy of change, change propagation analysis is performed within a WS-BPEL business process. An algorithm for change impact analysis is proposed.

The paper is structured as follows. Section 2 presents the business process model. Section 3 presents the change impact algorithm in WS-BPEL processes. Section 4 represents the case study. Section 5 presents an evaluation of the runtime results obtained. Section 6 represents conclusions.

2. Business process model

The service-oriented business process proposed is based on the model from [17]. The proposed model has two levels, the process level and the services level.These levels will be shortly presented next.

a) Process level

The process level contains business processes referred to as internal processes. An internal process consists of a control flow scheme and an information flow diagram.

Control flow scheme

The control flow scheme consists of a set of control activities and the relationships associated with them. The activities are classified into private activities (P-activities) and communication activities (C-activities) [18], [19]. Private activities are invisible to the partners. P-standard BPEL activities allowed are: assign, throw, wait, empty, exit, rethrow. Communication activities are used to exchange information with partners. C-standard BPEL activities allowed are: receive, reply, invoke.

Definition 1 *The control flow scheme* of an internal process is defined as a 3-tuple [17]

SCF = (A, C, E) where :

- $A = \{a_1, ..., a_n\}$ is a set of activities. For each $a \in A$, if a is a C-activity, $a.partner$ denotes a partner that a tries to interact with ;
- $C = \{\oplus split, \oplus join, \otimes split, \otimes join\}$ is a set of control connectors, where $\oplus$ represents AND connector, while $\otimes$ represents XOR connector;
- $E = \{e_1, ..., e_m\}$ is a set of arrows associated to activities and connectors.

Information flow diagram

The information flow diagram defines how data is transferred between activities. The information flow diagram is a key requirement for understanding the data dependency between activities and for analyzing the impact of changes. The diagram is similar to the information flow diagram of the data stream as defined in [18].

$D = \{d_1,...,d_n\}$ is a set of data elements associated with the internal process. Each activity has an input parameter and/or an output parameter. $InPar(a)$ and $OutPar(a)$. A data connection is defined as: $dc = (d, a, par, \mathrm{mod})$, where $d \in D, a \in A, par \in InPar(a) \cup OutPar(a), \mathrm{mod} \in \{read, write\}$.

Definition 2 (Information flow diagram) $SCF = (A, C, E)$ is the control flow scheme of an internal process [17]. The information flow diagram is the set of all data connections $DFI = \{dc_1,...,dc_m\}$.

Definition 3 ***(***Activities data dependency) $SCF = (A, C, E)$ is the control flow scheme of an internal process, $DFI = \{dc_1,...,dc_m\}$ the information flow diagram and $D = \{d_1,...,d_n\}$ is the set of data elements associated with the internal process. For each $a_i, a_j \in A$, a_i depends on a_j , $a_i \xrightarrow{D} a_j$, if $\exists dc_x, dc_y \in DFI$, so that $dc_x = (d, a_j, par_s, write)$, $dc_y = (d, a_i, par_t, read)$, where $d \in D, par_s = OutPar(a_j), par_t = InPar(a_i)$ and a_j precedes a_i in SCF. An internal process can be defined as a 2-tuple: $IP = (SCF, DFI)$.

b) Service level

Service level includes services that are part of an internal process. Each service is an exterior view of the internal process in terms of a partner. An observable behavior of a service is offered to a partner through the interface of a service [20].

Definition 4 (Service) A service [21] is defined as a 2-tuple $s = (O, T)$, where:

-$O = \{o_1,...,o_n\}$ is a set of operations. Each operation $o_i \in O$ is associated with a set of messages;

- $T \subseteq O \times O$ is a control relations between operations. Each transition $t = (o_i, o_j) \in T(o_i, o_j \in O)$ represents the invocation of operation o_j by o_i. o_i is the source of the transition, while o_j is the destination of the transition. For a transition $t \in T, c(t)$ is a constraint on t. Transition t fires immediately after the execution of the source operation. If $c(t) \neq \emptyset$, transition t is executed immediately after $c(t)$ becomes true.

3. Change impact analysis

This section presents the change impact patterns. The impact change model captures the effect of specific changes. These models allow the change impact analysis and a better understanding of the impact of a change and the development of effective and efficient reactions change. Further models for the impact of a change in a BPEL process will be introduced and then a function will be defined for determining the impact of changes.

Insert an activity

This model describes the insertion of an internal process activity. There have been identified two ways to insert an activity. These are:
(1) inserting a sequential activity, (2) inserting a parallel activity.

Deleting activities

This model describes the removal of activities that belong to the internal process. One activity or a sequence of activities can be removed.

Replacing an activity with another activity

This model is based on previous models presented and consists of deleting one or several activities and inserting an activity.

By determining the direct impact of changes, there are identified the activities from a BPEL process that will be affected by the application of change models described above. FuncDep is a function $FuncDep : SCF, DFI, A_{change} \longrightarrow A_{result}$ where SFC is the control flow scheme of an internal process and DFI is the information flow diagram, A_{change} is the sequence of activities $A_{change} = \{a_1,...,a_m\}$ involved in the change, function A_{result} is the output of FuncDep, and consists of a set of activities $A_{result} = \{a_1,...,a_n\}$ affected by the change A_{change}. Between any two activities $a_i \in A_{change}, a_j \in A_{result}$ data dependency relationships may exist in the form: $a_i \xrightarrow{D} a_j$ or $a_j \xrightarrow{D} a_i$.

Initial step
outputVariables - hashMap {*key* - variable name, *values* - nodes representing activities that have as output the variable determined by key}

inputVariables - hashMap {*key* - variable name, *values* - nodes representing activities that have as input the variable determined by key }
variablesMap - hashMap { *key* - variable name, values of type *variableUsage*}
variableUsage - {*inputs* - a set of nodes for which the variable is an input variable, *outputs* - a set of nodes for which the variable is an output variable }

```
for each node $n_i$ of the tree SCF from start to end:
    $predecessors(n_i) = predecessors(n_{i-1}) \cup n_{i-1}$
    if $n_i$ is an activity
        if $n_i$ is an activity
        $\mathrm{var}iable\_v_{out} = output\_\mathrm{var}iable\_of\_n_i$
        $\mathrm{var}iable\_v_{in} = \mathrm{int}\,put\_\mathrm{var}iable\_of\_n_i$
        $\mathrm{var}iableUsage\_v_u$
        if $v_{in} \neq null$
            $if\_\mathrm{var}iablesMap.contains(v_{in})$
                $vu = \mathrm{var}iablesMap.get(v_{in})$
                $vu.addInput(v_{in})$
            else
                $vu.addInput(v_{in}); \mathrm{var}iablesMap.put(v_{in}, vu)$
        if $v_{out} \neq null$
            $if\_\mathrm{var}iablesMap.contains(v_{out})$
                $vu = \mathrm{var}iablesMap.get(v_{out})$
                $vu.addOutput(v_{out})$
            else
                $vu.addOutput(v_{out}); \mathrm{var}iablesMap.put(v_{out}, vu)$
```

Function FuncDep_delete_activities
Input: SCF, DFI, A_{change}, *variablesMap*
Output: A_{result}
$A_{change} = \{a_1, ..., a_m\}$ is a sequence of activities involved in the change

```
for each $a_i \in A_{change}$
    $\mathrm{var}iable\_v_{out} = output\_\mathrm{var}iable\_of\_a_i$
    if $v_{out} \neq null$
```

```
        $A_{dependents} = \{a_1,...,a_d\}$ - activities that have as input variable $v_{out}$
        $A_{dependents} = \mathrm{var}iablesMap.get(v_{out})$
        for each $a_j \in A_{dependents}$
            if $a_i \in predecessors(a_j)$
            $A_{result} = A_{result} \cup a_j$
```

Function FuncDep_add_activity
Input: SCF, DFI, A_{change}, *variablesMap*
Output: A_{result}

```
$A_{change} = \{a_1,....,a_m\}$ is a sequence of activities involved in the change
for each $a_i \in A_{change}$
    $\mathrm{var}iable\_v_{in} = input\_\mathrm{var}iable\_of\_a_i$
    $\mathrm{var}iable\_v_{out} = output\_\mathrm{var}iable\_of\_a_i$
    if $v_{in} \neq null$
        $A_{dependentsIn} = \{a_1,...,a_d\}$ - activities that have as output variable $v_{in}$
        $A_{dependentsIn} = \mathrm{var}iablesMap.get(v_{in})$
        for each $a_j \in A_{dependentsIn}$
            if $a_j \in predecessors(a_i)$
            $A_{result} = A_{result} \cup a_j$
    if $v_{out} \neq null$
        $A_{dependentsOut} = \{a_1,...,a_d\}$ - activities that have as input variable $v_{out}$
        $A_{dependentsOut} = \mathrm{var}iablesMap.get(v_{out})$
        for each $a_j \in A_{dependentsOut}$
            if $a_i \in predecessors(a_j)$
            $A_{result} = A_{result} \cup a_j$
```

Function FuncDep_replace_activity
Input: SFC, DFI, A_{delete}, A_{add}, *variablesMap*
Output: A_{result}

A_{result} = *FuncDep_delete_activity(SCF, DFI,* A_{delete}*, variablesMap)* $\cup$ *FuncDep_add_activity(SCF, DFI,* A_{add}*, variablesMap)*

If an activity involved in the change is of type C-activity, based on that activity's partnerLink, it is identified the BPEL based process that communicates with that activity. The algorithm involves the initial construction of the list of predecessors for each node of the tree, and variablesMap's hashMap containing the variables and the activities that have as input / output these variables, during the first traversal of the tree. The first traversal of the tree is realized when the XML containing the serialized BPEL based process is parsed using a SAX parser.

When one of the functions: *FuncDep_add_activity*, *FuncDep_delete_activity*, *FuncDep_replace_activity* is invoked, a list of predecessors is already built. The functions identify the tree nodes (activities) that have as input / output variable the input / output variable of an activity $a_i \in A_{change}$. If there is a direct path in the SCF tree between the two nodes (activities), then the identified activity will be affected by the activity $a_i \in A_{change}$ that is deleted / added.

One difference between the current approach and the approach proposed by Wang [17] is that the current approach is particularized for BPEL based processes, while the approach proposed by Wang is directed towards more general service-based processes. The complexity of the proposed algorithm for addition / deletion / replacement of activities is linear $\Theta(n)$. When the parsing of the XML file representing the BPEL based process is done, the whole tree of activities that compose the process is built. At the same time the list of predecessors for each activity is built and the input and output variables for each activity are determined and stored in Java classes of type java.util.HashMap. The complexity of adding/extracting of an element from a java.util.HashMap class depends on the implementation but is usual constant $\Theta(1)$. The worst case complexity possible for this operation is $\Theta(n)$. The change impact analysis complexity is determined by iterating the set of activities involved in the change equal to $\Theta(n)$ and the complexity of the required addition / extraction of input / output variable from the classes of type java.util.HashMap equal in the worst case with $\Theta(n)$. In the approach proposed in this paper the overall complexity will be equal to the sum of the two stated complexities. As constants are removed in the calculation of the complexity, the complexity of our change impact analysis algorithm will be equal to $\Theta(n)$.

In the approach proposed by Wang, in order to determine the impact of changes, for each current activity involved in the change, the algorithm iterates through all the process activities to detect the data dependencies between the current iterated activity and other activities that may be involved in the change. The total complexity is $\Theta(n^2)$, thus higher.

A comparison of execution time could not be made because Wang uses a database component to store processes and activities, while the current BPEL based process is represented by a file in XML format.

4. Case study and implementation

The implementation was done using Java SE 1.7. Eclipse Juno was chosen as the development environment. Graphics were done using Java Swing . When loading a BPEL based process exported as an XML, the document is parsed using a SAX parser . The parsing of the XML file is based on the tags defined in WS-BPEL 2.0 standard, During the parsing, a tree representing the BPEL process is constructed. Each node in the tree is associated with a list of predecessors corresponding to tree nodes in the path from the start of the process to the current node. For each node in the tree represented as an activity, its output and input variables v_{out} and v_{in} are added to the variablesMap together with the node (activity) name .

When the deletion / replacement of activities is desired, the first and last activity are selected from the tree. When the insertion of a sequential activity is desired, there is selected from the tree the activity after which the insertion will take place. When the insertion of a parallel activity is desired the first and the last activity for the insertion will be selected from the tree.

The impact of changes will be computed based on the presented algorithms and printed into a textbox.

For the case study, we have chosen a scenario where a significant portion of the dam of an industrial water pond breaks. Through the gap created 150,000 cubic meters of water resulting from cyanide process of gold extraction by cyanidation leak. Cyanide water gets into the valley located in the vicinity of the pond, getting into a river that passes through the valley. A more comprehensive description of the case study can be found in [22].

Governmental organization EmergGov partners up with other organizations(hospitals, emergency response organizations, organizations dealing with food and water distribution, organizations that monitor water, air and soil quality) and forms a virtual organization that will deal with the effects of the natural disaster. Several processes corresponding to different members of the virtual organization have been defined using the WS-BPEL modeling language.

The orchestrator is represented by the organization EmergGov „Emergency response in abnormal situations process". The pollutants monitoring organizations and the weather forecast stations are represented by web-services. Only three relevant processes for our case study are briefly described.

The "*Emergency response in abnormal situations*" process describes the reaction of EmergGov in a "disaster situation". Meteorological information are obtained from the weather forecast stations: air temperature, humidity, wind speed, precipitations level through the usage of: invokeMeteoService1, invokeMeteoService2, invokeMeteoService3. Pollutants level for air, water and soil in a certain area are obtained thorough the usage of: invokeWaterPollutantsMonitoringService, invokeAirPollutantsMonitoringService, invokeSoilPollutantsMonitoringService. Based on the information obtained concerning the pollutant level in a certain area, decisions are taken in the “if” block. If the pollutants level is normal in a certain area the else branch is executed and the pollutants values are stored in a database: invokeUpdateDatabase. If the pollutants level exceeds a certain limit, organizations that are specialized in fast interventions are contacted through the use of web-services: invokeWaterPollutionInterventionService, invokeAirPollutionInterventionService, invokeSoilPollutionInterventionService. Organizations that are specialized in food and water distribution are contacted through the use of invokeDistributionService1 and invokeDistributionService2. The SearchIdentifyDiagnosePersons process is invoked through invokeSearchIdentifyDiagnosePersons and the database is updated in this case using the pollutants values detected and identification numbers of the persons that needed medical assistance (Figure 1).

The “*Search Identify Diagnose Persons*” process receives as input data the name of the zone affected by the abnormal situation. Persons with problems will be searched. An initial diagnostic will be established, and if they need medical care, the EmergencyCare process will be invoked through invokePatientEmergencyCare. The process returns the identification number of a person that needs medical attention.

In the “*Emergency Care Process*” process, the medical record of a patient is obtained using its personal identification number (invokeGetPatientMedicalRecord). Medical tests are harvested for the patient (invokeHarvestingMedicalTests) and analyzed (invokeMedicalTestsAnalysis). Based on the results from the analysis of the medical tests, a diagnostic is established (invokePatientDiagnose). Based on the severity of the diagnostic a decision is taken in the “if” block. The patient can be hospitalized (invokeHospitalization), transferred to another unit (invokePatientTransferToAnotherUnit) or discharged (invokePatientDischarge).

As we can see from figure 1, the understanding of the logic of BPEL processes is pretty difficult without comprehensive explanations. The activities that compose the BPEL process do not contain a description and the variables used are not easy accessible. The understanding of the changes impact is not easy without a thorough previous understanding of the process. The change impact analysis tool addresses this problem. Change impact analysis for the insertion/deletion/replacement of activities is made.

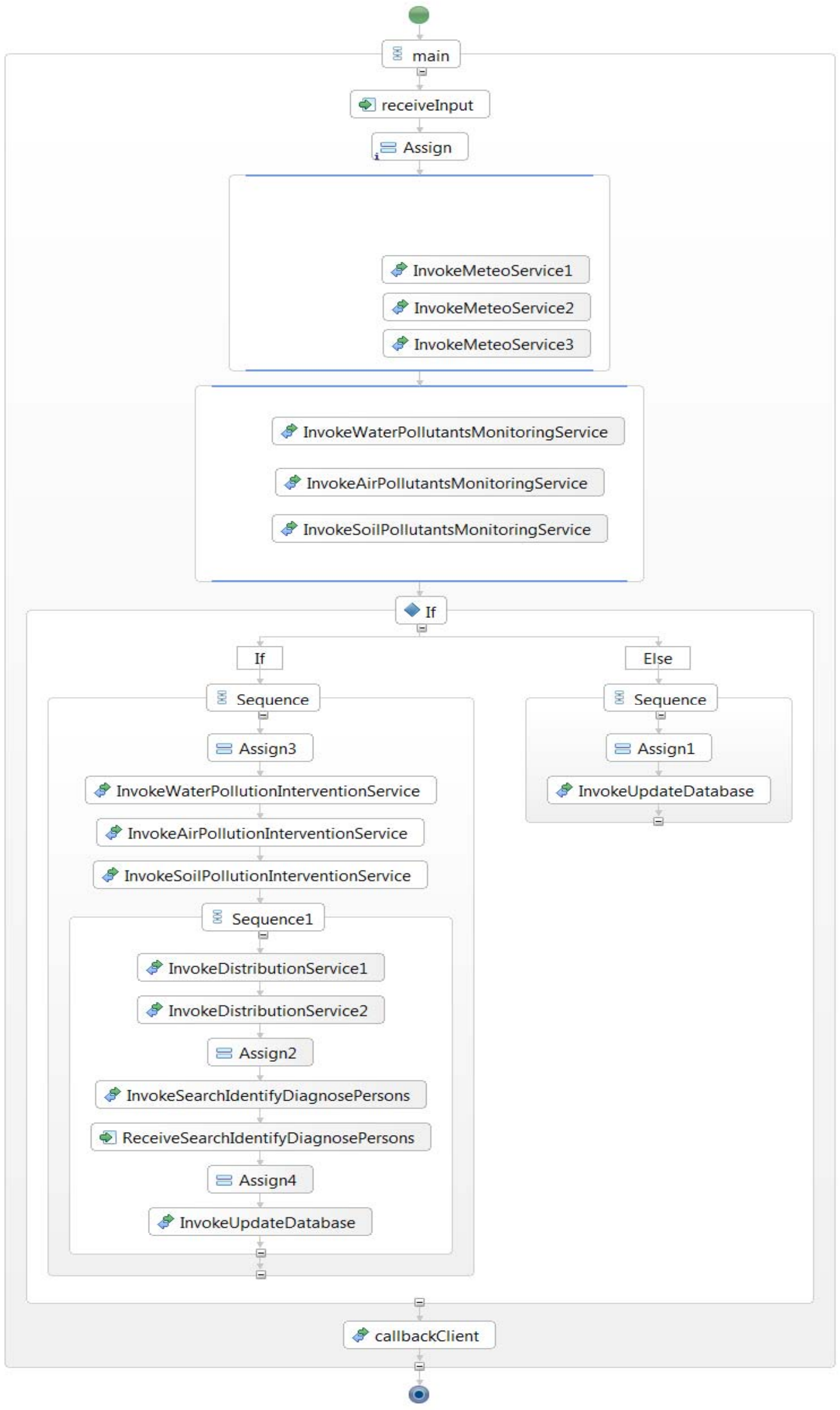

main
receiveInput
Assign
InvokeMeteoService1
InvokeMeteoService2
InvokeMeteoService3
InvokeWaterPollutantsMonitoringService
InvokeAirPollutantsMonitoringService
InvokeSoilPollutantsMonitoringService
If
If
Else
Sequence
Assign3
InvokeWaterPollutionInterventionService
InvokeAirPollutionInterventionService
InvokeSoilPollutionInterventionService
Sequence1
InvokeDistributionService1
InvokeDistributionService2
Assign2
InvokeSearchIdentifyDiagnosePersons
ReceiveSearchIdentifyDiagnosePersons
Assign4
InvokeUpdateDatabase
Sequence
Assign1
InvokeUpdateDatabase
callbackClient

Fig. 1. Emergency response in abnormal situations process

1) In the “Assign” activity the request(input) variables for the communication activities invokeWaterPollutantsMonitoringService, invokeAirPollutantsMonitoringService, invokeSoilPollutantsMonitoringService are initialized. The invocation of these services is made for a certain area, as described above. Because we do not want anymore to statically initialize this variables, we decide to delete this activity and replace it with a receive activity. The receive activity will have as output the WaterPollutantsMonitoringServicePartnerLinkRequest variable. The effect of the deletion of the Assign1 activity seems to involve only invokeWaterPollutantsMonitoringService, invokeAirPollutantsMonitoringService, invokeSoilPollutantsMonitoringService, but according to the change impact tool analysis the following activities are affected: *InvokeWaterPollutantsMonitoringService; InvokeAirPollutantsMonitoringService; Assign3; InvokeSoilPollutantsMonitoringService;* The adding of a receive activity that has as output variable WaterPollutantsMonitoringServicePartnerLinkRequest affects the following activities: *InvokeWaterPollutantsMonitoringService; Assign3.*

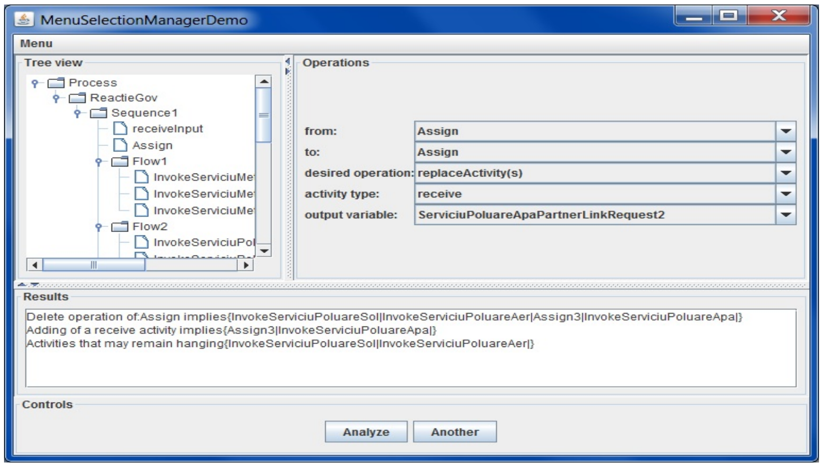
We can observe that *InvokeAirPollutantsMonitoringService* and *InvokeSoilPollutantsMonitoringService* remain hanging (Figure 2).

2) The organization that measures water pollutants level (*InvokeWaterPollutantsMonitoringService*) decides to leave the VO. In order to determine the impact of this change the *InvokeWaterPollutantsMonitoringService* activity is deleted. The affected activities are: *If; Assign4; Assign1*

3) We suppose that we do not want anymore to invoke the Search Identify Diagnose Persons Process from within the Emergency response in abnormal situations process. The deletion of the *invokeSearchIdentifiyDiagnosePersons* activities will not affect the current process but we can see that the process impacted is Search Identify Diagnose Persons Process referred in the tool as “CautareIdentificareSalvarePersoane”.

5. Runtime results evaluation and discussions

The tool was tested on two computers and the execution time was measured (Table 1 and 2). The comparable execution times for change impact analysis (Table 2) on the used computers suggest that the change impact analysis is optimally implemented.



from: Assign
to: Assign
desired operation: replaceActivity(s)
activity type: receive
output variable: ServiciuPoluareApaPartnerLinkRequest2

```
Delete operation of:Assign implies{InvokeServiciuPoluareSol|InvokeServiciuPoluareAer|Assign3|InvokeServiciuPoluareApa|}
Adding of a receive activity implies{Assign3|InvokeServiciuPoluareApa|}
Activities that may remain hanging{InvokeServiciuPoluareSol|InvokeServiciuPoluareAer|}
```

Fig. 2 Change impact when the Assign activity is deleted

Table 1

Load time for BPEL files

Load time (seconds)	Emergency Care	Search Identify Diagnose Persons	Patient Discharge	Patient Hospitalization	Emergency response	Patient Transfer
Intel core i3, 4Gb ram	0.073330156	0.007617414	0.009855315	0.009880972	0.046826571	0.01052811
Intel core i7, 32Gb ram	0.047350924	0.003969679	0.004948522	0.005674209	0.012383279	0.006096279

Table 2

Measured time for different operations

Operations time (seconds)	1) Replace activity	2) Delete activity	3) Delete activity	4) Add parallel activity	5)Add sequential activity
i3, 4Gb ram	0.002146104	0.000879766	0.000900862	0.000807925	0.000710997
i7, 32Gb ram	0.002283538	0.000616213	0.000625621	0.000635885	0.000489208

6. Conclusions

The proposed tool for change impact analysis identifies the activities affected by the operations of deletion/addition of activities and preconditions necessary. Also, if the communication activities invoke other BPEL processes, these processes are identified by name and displayed. By applying the change impact analysis tool for the WS-BPEL processes defined for the case study the advantages identified were: immediate awareness in real time, the impact analysis of a change within the WS- BPEL web service based processes.

Acknowledgment
The work has been funded through the European Social Fund, Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection, Financial Agreement POSDRU/107/1.5/S/76813.

R E F E R E N C E S

[1] Future Internet Enterprise Systems (FInES) Research Roadmap, http://cordis.europa.eu/fp7/ict/enet/documents/fines-researchroadmap-final-report.pdf , June 2010
[2] *T. Earl,"* SOA Principles of Service Design", Prentice Hall, 2008
[3] Web Service Description Language(WSDL) 1.1, http://www.w3.org/TR/wsdl
[4] Simple Object Access Protocol (SOAP) 1.2, http://www.w3.org/TR/soap/
[5] Web Services Business Process Execution Language Version 2.0 (WS-BPEL), http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf
[6] *F. Curcubera, M. Duftler, R. Khahaf, W. Nagy, N. Mukhi, S. Weerawarana*: Unravelling the Web Services Web: An introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing 6(2), 2002, pp. 86-93
[7] *M. Papazoglu, K. Pohl, M. Parkin, A. Metzger*, "Service Research Challenges and Solutions for Future Internet", Springer-Vertag Berling Heidelberg, 2010
[8] Modelio BPEL Designer, http://archive.modeliosoft.com/en/modules/modelio-bpel-designer.html, retrieved October 2013
[9] Eclipse BPEL designer, http://www.eclipse.org/bpel, retrieved October 2013
[10] http://www.oracle.com/technetwork/middleware/soasuite/downloads/index.html, 2013
[11] http://www.oracle.com/technetwork/middleware/bpel/overview/ds-bpel-11gr1-1-134826.pdf, retrieved October 2013
[12] http://technology.amis.nl/2012/11/18/oracle-soa-suite-11g-ps-5-introduces-bpel-with-conditional-correlation-for-aggregation-scenarios/, retrieved October 2013
[13] http://www-01.ibm.com/support/docview.wss?uid=swg24030614, retrieved October 2013
[14] http://pic.dhe.ibm.com/infocenter/esbsoa/wesbv7r5/index.jsp?topic=%2Fcom.ibm.wbpm.%20wid.bpel.doc%2Ftopics%2Fcunact.html , retrieved October 2013
[15] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.bt ools.help.modeler.doc/doc/tasks/widmodeling/analyzingchanges.html, retrieved Oct 2013
[16] *M. P. Papazoglou*, "The challenges of service evolution," in *CAiSE*, Montpellier, France, 2008, pp. 1–15
[17] *Y. Wang, J. Yang, W. Zhao*: Change impact analysis for service based business processes, 2010
[18] M. *Dumas, B. Benatallah, and H. R. M. Nezhad*, "Web service protocols: Compatibility and adaptation," vol. 31, no. 3, 2008, pp. 40–44
[19] *M. Reichert and P. Dadam*, "Adeptflex-supporting dynamic changes of workflows without losing control," *J. Intell. Inf. Syst.*, vol. 10, no. 2, 1998, pp. 93–129
[20] *L. de Alfaro and T. Henzinger*, "Interface automata," in *joint ESEC/FSE*, 2001, pp. 109–120
[21] Business process execution language for web services version 1.1, 2003. http://www.ibm.com/developerworks/library/ws-bpel/, retrieved October 2013
[22] *C. Danila*, "Dynamic Configuration of Collaborative Networked Organization by „Internet of Services" technologies", Phd Thesis, 2013