Change Pattern-Driven Traceability of Business Processes

Watcharin Uronkarn and Twittie Senivongse

Abstract—**Business analysts define business process models for describing a series of activities to produce services or products to serve business goals. Hence business process models represent business requirements for development of the software that enables automation of the business processes. When activities in a business process are changed, such changes also trigger changes in the artifacts that have been produced during development of the related software. Analysis of an impact a business process change has on the software is useful for the software project leader and the system analyst to plan the effort to change the artifacts, including the software itself, accordingly. This paper proposes change pattern-driven traceability of a business process by maintaining traceability information of a current business process model and comparing such a model with the newly designed one that incorporates the changes. The comparison discovers change patterns and traceability allows the artifacts that are affected by such change patterns to be identified. We present a tool to support the proposed approach.**

Index Terms—**business process model, business process, traceability, change pattern, change impact**

I. INTRODUCTION

BUSINESS process modeling is an activity to capture processes of business applications into business process models. Business process models hence describe a series of activities to produce services or products to serve business goals under certain business rules. They are used for communicating business process information to all business users – from business analysts who create the models of processes, to technical developers who are responsible for implementing the technology that will perform the activities in those processes, and to business people who will manage and monitor those processes [1]. To visualize business processes, a business process model presents process information using a flow chart-like graphical notation such as the Business Process Model and Notation (BPMN) [1].

From a technical perspective of a software project team, a business process model captures business requirements for the development of software that enables automation of the business process. That is, it leads to development of several software artifacts such as requirement specifications, analysis and design models, implementation components, tests, and other project documents [2]. When activities in a business process model are changed, such changes also trigger changes in the artifacts related to those activities. Analysis of an impact a business process change has on the software is useful for the software project leader and the system analyst to plan the effort to change the artifacts, including the software itself, accordingly. For example, if the change has a small impact, change to software documents and code is preferable. On the other hand, it might be better to develop new software if the change affects the existing software to a large extent in such a way that only a small part can be reused. As a result, this paper addresses three requirements to manage business process change:

Requirement#1: To determine the impact of a change on a current business process, we need to make the current business process traceable. As software requirements traceability refers to the ability to describe and follow the life of a requirement in both forward and backward directions, information about the relationships between software requirements and many kinds of associated artifacts have to be maintained [2] so that the scope of the initiating change can be analyzed. This is called *traceability impact analysis* [3]. In our case, we need to extend traceability information to also document the relationships between activities in the current business process model and other kinds of software artifacts.

Requirement#2: To determine the impact of a business process change, we need to locate the change made to the current business process, determine the type of change, and identify the affected activities.

Requirement#3: As the business process and the change may be complex and there may be several associated software artifacts, informative information regarding the types of change and parts of the existing software that are impacted should be provided to the project leader and the system analyst so that they can determine subsequent changes that are to be made to complete the change in the business process. This information should be provided in a manner that is as automated as possible.

To answer to these requirements, this paper proposes change pattern-driven traceability of a business process together with a tool to support the proposed approach. We associate a current business process model with traceability information that links the model to other software artifacts, i.e., requirements, use cases, design classes, and programs. When a new version of the business process model is designed to incorporate changes in business requirements,

Manuscript received December 8, 2013 revised January 9, 2014.

Watcharin Uronkarn is with the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330 Thailand; email: watcharin.u@student.chula.ac.th

Twittie Senivongse is with the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Bangkok 10330 Thailand; corresponding author phone: +66 2 2186996; fax: +66 2 2186955; email: twittie.s@chula.ac.th

the two versions are compared to identify the patterns (or types) of change, affected business process activities, and affected software artifacts.

The rest of this paper is organized as follows. Section II discusses related work and section III describes the approach which comprises business process traceability, change pattern detection, and change impact analysis. Section IV concludes the paper with a future outlook.

II. Related Work

Traceability impact analysis is a widely addressed issue in software requirements management. Literature has reported different approaches to enable traceability and the use of traceability information to analyze the scope and degree of requirement change impact at both code and design level of the software, by tracing the relationships between software artifacts. Here we focus on the impact of change that is not initiated at the software but at the business process level.

Piprani et al. [4] argue that, in reality, business requirements are generally surfaced over several years in memos, e-mails, meeting minutes, consultant reports etc. Usually these requirements are not documented and hence the requirements gathering effort for a software project has to start all over again to capture those already stated but undocumented requirements. They propose an Object Role Modeling (ORM) based metamodel for modeling traceability information across a multitude of documents and across development phases. Their approach is seen as an effort to extend traceability information to include documents other than typical software artifacts. Similarly, we extend traceability to the business process level and allow tracing between a business process model and an analysis model of the software by documenting a relationship between an atomic activity (or a task) in a business process model and a use case in a use case diagram. This is the approach taken by IBM's Rational System Architect for transforming a BPMN diagram to a UML use case diagram [5].

A number of researchers have tackled the problem of change impact analysis for business processes. For example, Wang et al. [6] define change types and change impact patterns for business processes that involve invocation to Web services. Their change types are process change and service change. Specifically, process change consists of change types such as insert/remove/move an activity and replace/parallelize/sequence activities etc. Change impact patterns define how changes in the internal activities of the process and changes in the external services are propagated to the business process. Similarly, Xiao et al. [7] define a number of service change types and their business component impact set that signifies a set of tasks in the business process which are affected by a particular service change. Unlike these approaches, ours targets business process change at the modeling level, not the execution level, and our impact set will comprise the software artifacts that relate to the changed process model.

In addition, change types such as those by [6] and [7] are rather primitive and not so informative for the system analyst to reason about business requirements change. We choose to adopt Dijkman's classification of structural differences between business processes [8] as the patterns of change between two versions of a business process model. For a complete detail, refer to [8]; here we summarize the classification as follows:

1) *Authorization differences*: An activity in one process is assigned to different roles in the other process, i.e., *different roles*, *single role vs. collection of roles*, and *different collections of roles*.

2) *Activity differences*: A collection of activities in one process corresponds to a different collection of activities, or not at all, in the other process, i.e., *skipped activity*, *interchanged activities*, *refined activities*, *corresponding collections of activities*, and *partly corresponding (collections of) activities*.

3) *Control flow differences*: Control flow relations between a collection of activities in one process are different from those between an equivalent collection of activities in the other process, e.g., *different dependencies*, *additional dependencies*, *activities occur at different moments in processes*, *iterative vs. once-off occurrence*.

III. Change Pattern-Driven Traceability

The process of change pattern-driven traceability of business processes is depicted in Fig. 1. A business analyst who designs a business process model can use our supporting tool to perform each step of this process.

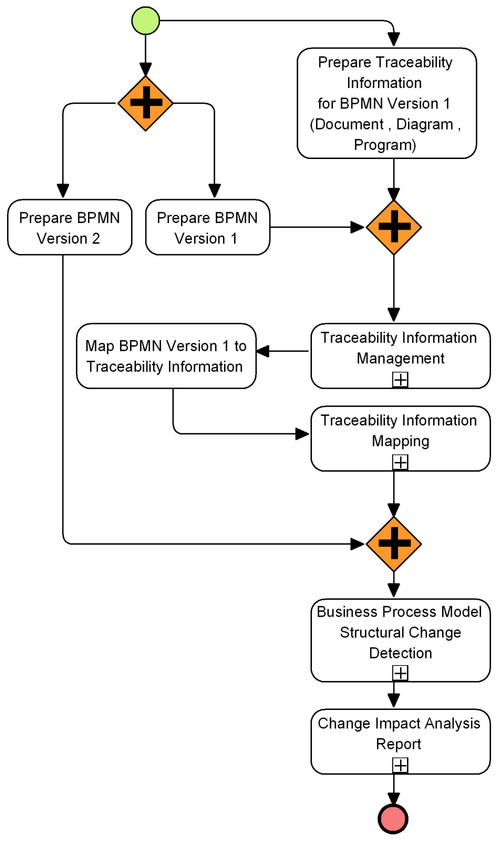

Prepare Traceability Information for BPMN Version 1 (Document , Diagram , Program)
Prepare BPMN Version 2
Prepare BPMN Version 1
Map BPMN Version 1 to Traceability Information
Traceability Information Management
Traceability Information Mapping
Business Process Model Structural Change Detection
Change Impact Analysis Report

Fig. 1. Overview of change pattern-driven traceability.

We assume that, first there were business requirements and a business analyst designed a business process model (i.e., *BPMN version 1*) whose activities answered to the requirements. Based on this BPMN version 1, artifacts were produced during the software development process, including the application program. Therefore, to enable traceability, the project leader and the system analyst will have to define mapping between the business process model and associated software artifacts.

Later when there are changes in the business process, the business analyst designs a new version of the business process model (i.e., *BPMN version 2*) to reflect the changes. The question raised by the software project team is, to what extent these changes will affect the associated application? Hence, the two versions of the business process model will be compared so that the change patterns can be detected and parts of the business process model and software artifacts that may need modification can be reported.

The rest of this section explains each step in detail.

A. Traceability Information Management

Using the supporting tool, first the business analyst imports the BPMN version 1 in XML format. (Note that, the XML representation of a BPMN business process model can be obtained from a BPMN tool such as Visual Paradigm.) An under-warranty after-sales service of an agricultural machinery company in Thailand, as shown in Fig 2(a), exemplifies the BPMN version 1.

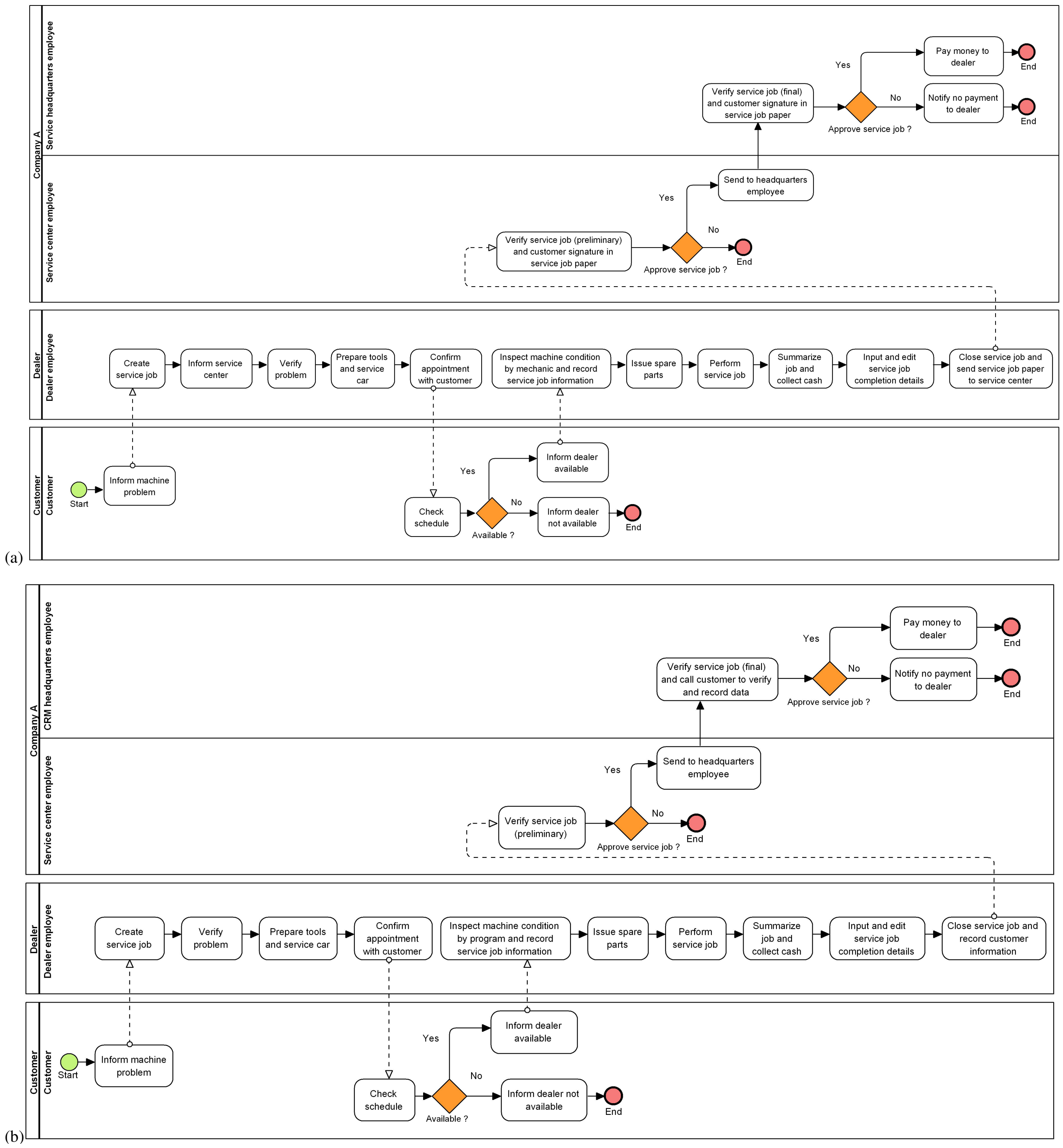
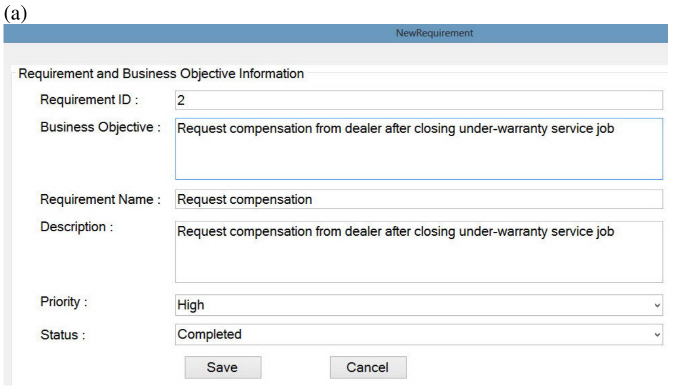

Company A
Service headquarters employee
Pay money to dealer
End
Yes
Verify service job (final) and customer signature in service job paper
No
Notify no payment to dealer
End
Approve service job ?
Service center employee
Send to headquarters employee
Yes
No
Verify service job (preliminary) and customer signature in service job paper
End
Approve service job ?
Dealer
Dealer employee
Create service job
Inform service center
Verify problem
Prepare tools and service car
Confirm appointment with customer
Inspect machine condition by mechanic and record service job information
Issue spare parts
Perform service job
Summarize job and collect cash
Input and edit service job completion details
Close service job and send service job paper to service center
Customer
Customer
Start
Inform machine problem
Yes
Inform dealer available
Check schedule
No
Inform dealer not available
End
Available ?
(a)
Company A
CRM headquarters employee
Pay money to dealer
End
Yes
Verify service job (final) and call customer to verify and record data
No
Notify no payment to dealer
End
Approve service job ?
Service center employee
Send to headquarters employee
Yes
Verify service job (preliminary)
No
End
Approve service job ?
Dealer
Dealer employee
Create service job
Verify problem
Prepare tools and service car
Confirm appointment with customer
Inspect machine condition by program and record service job information
Issue spare parts
Perform service job
Summarize job and collect cash
Input and edit service job completion details
Close service job and record customer information
Customer
Customer
Start
Inform machine problem
Yes
Inform dealer available
No
Check schedule
Inform dealer not available
End
Available ?
(b)

Fig. 2. Under-warranty after-sales service business process model (a) version 1 (b) version 2.

This business process model depicts the steps a dealer takes to service a machine repair job under warranty. The dealer does not charge the customer for the repair but the dealer can request compensation from the headquarters for all labor and spare parts costs.

To make the BPMN version 1 traceable, the project leader and the system analyst provide the artifact information that is relevant to the software that automates it. This includes (1) requirement and business objective, (2) use case in a use case diagram, (3) functional requirement in a software requirement specification, (4) class in a class diagram, and (5) program. An example of traceability information (i.e., a requirement named *Request compensation*) related to the BPMN version 1 is shown in Fig. 3.

(a)

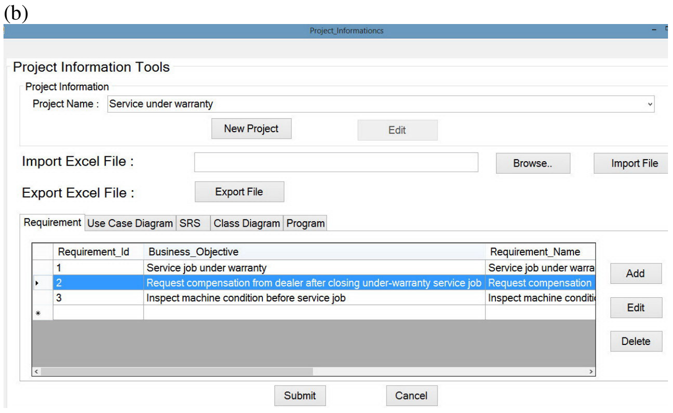


NewRequirement

Requirement and Business Objective Information

Requirement ID :	2
Business Objective :	Request compensation from dealer after closing under-warranty service job
Requirement Name :	Request compensation
Description :	Request compensation from dealer after closing under-warranty service job
Priority :	High
Status :	Completed

(b)



Project_Informationcs

Project Information Tools

Project Information

Project Name : Service under warranty

Import Excel File :

Export Excel File :

Requirement_Id	Business_Objective	Requirement_Name
1	Service job under warranty	Service job under warra
2	Request compensation from dealer after closing under-warranty service job	Request compensation
3	Inspect machine condition before service job	Inspect machine conditi

Fig. 3. Example of traceability information (a) adding information (b) list of recorded information.

B. Traceability Information Mapping

Using XML DOM parser, the supporting tool will extract the activities in the BPMN version 1 and other structural information (i.e., flow objects, data, connecting objects, and swimlanes). We assume that all activities are atomic activities (aka. tasks in BPMN) and each activity will be mapped to appropriate software artifacts so that the relationships can be traced between them. Fig. 4 presents the metamodel of traceability information mapping which extends traceability information by not only tracing software artifacts but also documenting the relationships between software artifacts and the activities (or tasks) in the BPMN version 1 (i.e., *Requirement#1*). A task in the BPMN version 1 can be traced backward to a requirement and business objective, and forward to a use case which in turn can be traced further to a functional requirement in a software requirement specification and also to a class in a class diagram, followed by the program code by which the class is implemented. An example of a mapping that traces a task in the BPMN version 1 (i.e., a task named *Close service job and send service job paper to service center*) to all relevant artifacts (e.g., requirements named *Service job under warranty* and *Request compensation*) is shown in Fig. 5. The mapping information is stored in a traceability information mapping database.

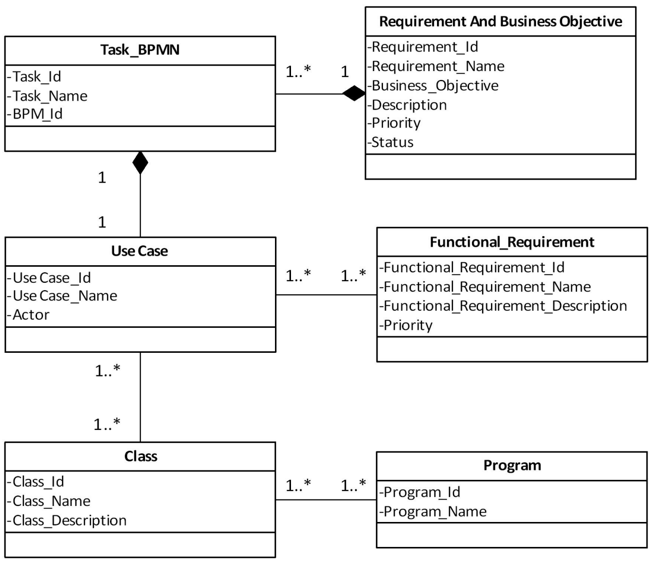
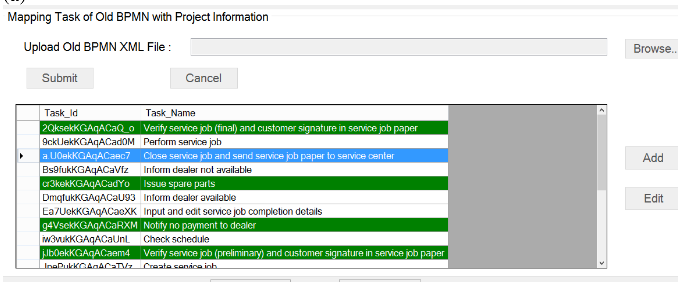


Fig. 4. Metamodel for traceability information mapping.

(a)

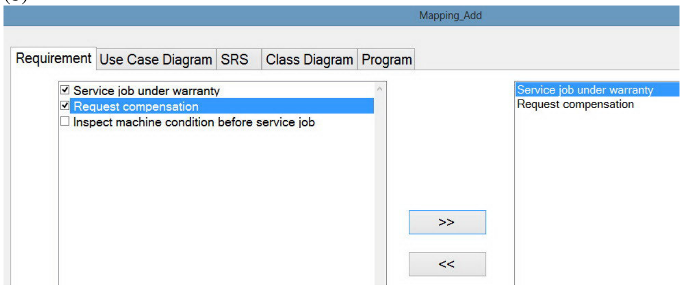


Mapping Task of Old BPMN with Project Information

Upload Old BPMN XML File :

Task_Id	Task_Name
2QksekKGAqACaQ_o	Verify service job (final) and customer signature in service job paper
9ckUekKGAqACad0M	Perform service job
a.U0ekKGAqACaec7	Close service job and send service job paper to service center
Bs9fukKGAqACaVfz	Inform dealer not available
cr3kekKGAqACadYo	Issue spare parts
DmqfukKGAqACaU93	Inform dealer available
Ea7UekKGAqACaeXK	Input and edit service job completion details
g4VsekKGAqACaRXM	Notify no payment to dealer
iw3vukKGAqACaUnL	Check schedule
jJb0ekKGAqACaem4	Verify service job (preliminary) and customer signature in service job paper
IpePukKGAqACaTVz	Create service job

(b)



Mapping_Add

☑ Service job under warranty
☑ Request compensation
☐ Inspect machine condition before service job

Service job under warranty
Request compensation

Fig. 5. Example of traceability information mapping (a) select task to be mapped (in blue) (b) select software artifacts to map them to selected task.

C. Business Process Model Structural Change Detection

When there are changes in the business process, the business analyst redesigns the business process model such as the BPMN version 2 in Fig. 2(b) and inputs its XML representation to the supporting tool. Its activities and structural information are extracted and compared to those of the BPMN version 1. The comparison can detect the changes made to the BPMN version 1, determine the types of changes, and identify the affected activities (i.e., *Requirement#2*).

For the two versions of the business process model in Fig.

2, the following change patterns are applied to a number of activities:

1) *Authorization differences – different roles* is applied to the activities *Pay money to dealer* and *Notify no payment to dealer* as the performing role is changed from Service headquarters employee in the BPMN version 1 to CRM headquarters employee in the BPMN version 2.

2) *Activity differences – skipped activities* is applied to the activity *Inform service center* in the BPMN version 1.

3) *Activity differences – interchanged activities* is applied to the activity *Inspect machine condition by mechanic and record service job information* in the BPMN version 1 as it is equivalent to or has the same effect as the corresponding activity *Inspect machine condition by program and record service job information* in the BPMN version 2.

4) *Activity differences – corresponding collections of activities* is applied to the following four activities in the BPMN version 1: *Close service job and send service job paper to service center*, *Verify service job (preliminary) and customer signature in service job paper*, *Send to headquarters employee*, and *Verify service job (final) and customer signature in service job paper*. These activities altogether are equivalent to or produce the same effect as the following four corresponding activities in the BPMN version 2: *Close service job and record customer information*, *Verify service job (preliminary)*, *Send to headquarters employee*, and *Verify service job (final) and call customer to verify and record data*.

5) *Control flow differences – different dependencies* is relevant to the following five activities in the BPMN version 1 because the sets of preceding activities in the two BPMN versions on which they depend are different: *Verify problem*, *Issue spare parts*, *Send to headquarters employee*, *Pay money to dealer*, and *Notify no payment to dealer*. In this example, these five activities form a natural impact set of other change patterns, e.g. *Verify problem* has different dependencies as its preceding activity is changed from *Inform service center* in the BPMN version 1 to *Create service job* in the BPMN version 2 because *Inform service center* is skipped.

The developed tool supports the detection of 12 of Dijkman's change patterns previously mentioned in Section II. When comparing the two versions of the business process, the tool applies change pattern detection algorithms. Most of these change patterns can be detected automatically, while four of them cannot (i.e., *interchanged activities*, *refined activities*, *corresponding collections of activities*, and *partly corresponding (collections of) activities*) since the semantic knowledge about the activities is required. For these latter cases, the tool works in a semi-automatic manner, i.e., the business analyst has to guide the detection by specifying corresponding activities between the two versions. Due to space limitation, here we summarize the detection algorithms that are used to find the change patterns in our example as above.

Detection Algorithm for Authorization Differences – Different Roles

1) Check to see that an activity exists in both BPMN version 1 and BPMN version 2.

2) Check to see that this activity is performed by a single role in each version.

3) Check to see if the performing roles for this activity are different in both versions. If so, it is the case of the different roles pattern.

Detection Algorithm for Activity Differences – Skipped Activities

1) Check to see that an activity in the BPMN version 1 does not exist in the BPMN version 2.

2) Check to see that the closest preceding and succeeding activities of this activity exist in the BPMN version 2.

3) Check to see if those closest preceding and succeeding activities are directly connected in the BPMN version 2. If so, it is the case of the skipped activities pattern.

Detection Algorithm for Activity Differences – Interchanged Activities

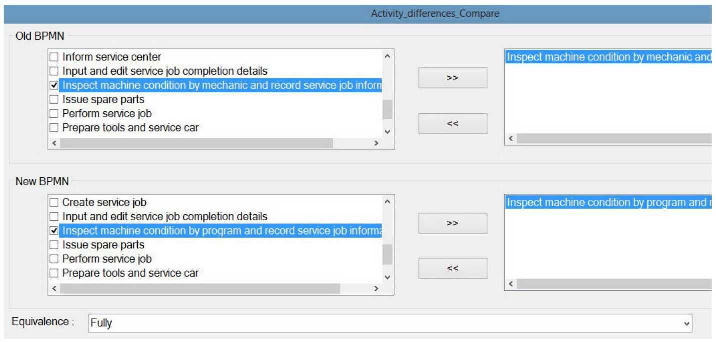
1) Check for the activities in the BPMN version 1 which do not exist in the BPMN version 2.

2) Receive input from the business analyst which specifies that a set of activities in the BPMN version 1 is equivalent to a set of activities in the BPMN version 2.

3) Receive input from the business analyst which specifies that the corresponding set of activities in the BPMN version 2 is fully or partly equivalent to the set of activities in the BPMN version 1.

4) Check to see if each of the two sets contains a single activity and the activities are fully equivalent. If so, it is the case of the interchanged activities pattern.

An example of interchanged activities is shown in Fig. 6. The business analyst selects from the lists of activities that cannot find corresponding activities in the other version of the BPMN model. It is specified that the activity *Inspect machine condition by mechanic and record service job information* in the BPMN version 1 is interchanged with and fully equivalent to the activity *Inspect machine condition by program and record service job information* in the BPMN version 2.


Activity_differences_Compare

Old BPMN

- [ ] Inform service center
- [ ] Input and edit service job completion details
- [x] Inspect machine condition by mechanic and record service job inform
- [ ] Issue spare parts
- [ ] Perform service job
- [ ] Prepare tools and service car

Inspect machine condition by mechanic and

New BPMN

- [ ] Create service job
- [ ] Input and edit service job completion details
- [x] Inspect machine condition by program and record service job informa
- [ ] Issue spare parts
- [ ] Perform service job
- [ ] Prepare tools and service car

Inspect machine condition by program and r

Equivalence : Fully

Fig. 6. Example of interchanged activities specified by business analyst.

Detection Algorithm for Activity Differences – Corresponding Collections of Activities

1) Check for the activities in the BPMN version 1 which do not exist in the BPMN version 2.

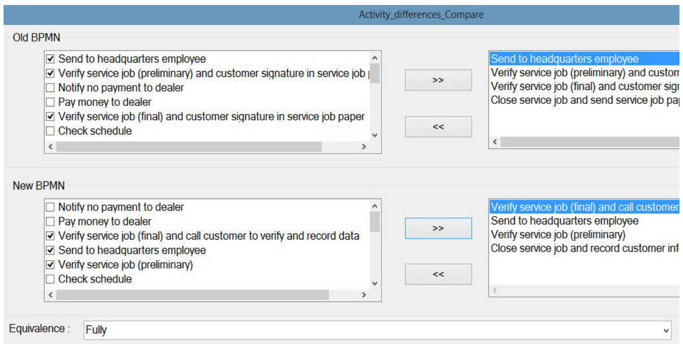
2) Receive input from the business analyst which specifies that a set of activities in the BPMN version 1 is equivalent to a set of activities in the BPMN version 2.

3) Receive input from the business analyst which specifies

that the corresponding set of activities in the BPMN version 2 is fully or partly equivalent to the set of activities in the BPMN version 1.

4) Check to see if the set of activities in the BPMN version 1 contains multiple activities and they are fully equivalent to the corresponding set of activities in the BPMN version 2. If so, it is the case of the corresponding collections of activities pattern.

An example of corresponding collections of activities is shown in Fig. 7. The business analyst specifies that the task performed by four activities in the BPMN version 1 is fully equivalent to the task performed together by four different activities in the BPMN version 2.



Activity_differences_Compare

Old BPMN

- [x] Send to headquarters employee
- [x] Verify service job (preliminary) and customer signature in service job
- [ ] Notify no payment to dealer
- [ ] Pay money to dealer
- [x] Verify service job (final) and customer signature in service job paper
- [ ] Check schedule

Send to headquarters employee
Verify service job (preliminary) and custom
Verify service job (final) and customer sigr
Close service job and send service job pa

New BPMN

- [ ] Notify no payment to dealer
- [ ] Pay money to dealer
- [x] Verify service job (final) and call customer to verify and record data
- [x] Send to headquarters employee
- [x] Verify service job (preliminary)
- [ ] Check schedule

Verify service job (final) and call customer
Send to headquarters employee
Verify service job (preliminary)
Close service job and record customer inf

Equivalence : Fully

Fig. 7. Example of corresponding collections of activities specified by business analyst.

Detection Algorithm for Control flow differences – different dependencies

1) Check to see that an activity exists in both BPMN version 1 and BPMN version 2.

2) Check to see that this activity has the same number of closest preceding activities in both versions.

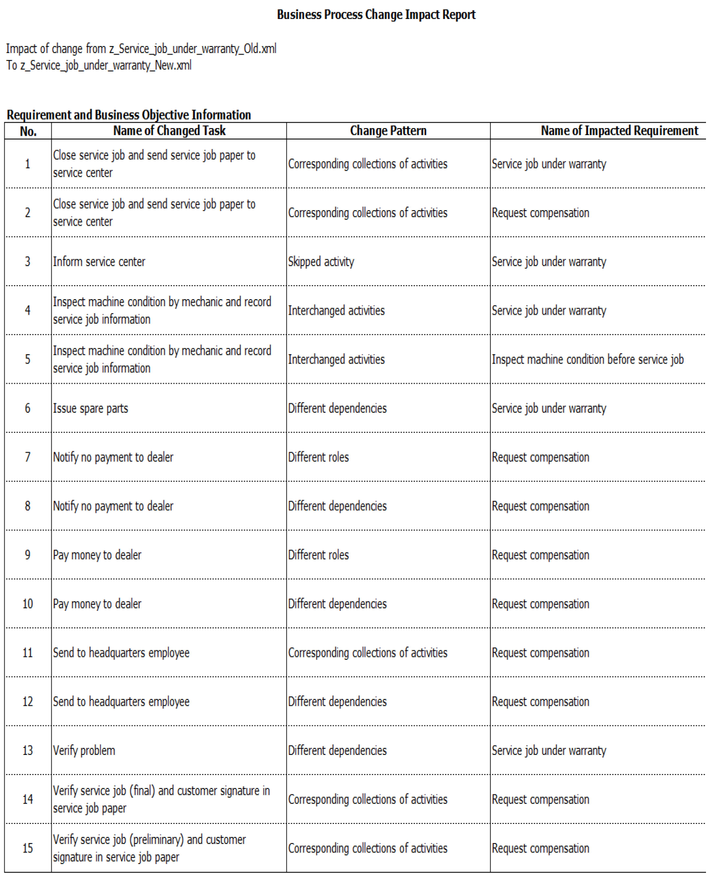
3) Check to see if the sets of closest preceding activities in both versions are different. If so, it is the case of the different dependencies pattern.

D. *Change Impact Analysis Report*

When change patterns are identified, the activities to which the change patterns are applied form an impact set, i.e., a set of impacted activities. From each impacted activity, the supporting tool traces its traceability information mapping to its associated software artifacts to analyze further impact. The tool reports the impacted activities in the BPMN version 1, change patterns that are applied to them, and each kind of software artifacts that are affected (i.e., *Requirement#3*). An example of the change impact report for the activities in the BPMN version 1 with regard to the requirement and business objective artifact is shown in Fig. 8. The project leader and the system analyst may use the report to guide them through the evaluation of the impact scope and planning for the change.

IV. Conclusion

The proposed approach uses the business process change patterns that exist between two versions of a business process model to drive the traceability impact analysis in the presence of business process change. Traceability information is extended to record the relationships between software artifacts and a business process activity that originates them. The developed tool can detect change patterns and report the impacted activities and artifacts. However, detection of some change patterns cannot be fully automated as semantic knowledge about the activities is required. To improve the situation, we can adopt techniques to determine semantic similarity between activities. We can also have the impact analysis results visualized to make the analysis report more intuitive.



Business Process Change Impact Report

Impact of change from z_Service_job_under_warranty_Old.xml
To z_Service_job_under_warranty_New.xml

Requirement and Business Objective Information

No.	Name of Changed Task	Change Pattern	Name of Impacted Requirement
1	Close service job and send service job paper to service center	Corresponding collections of activities	Service job under warranty
2	Close service job and send service job paper to service center	Corresponding collections of activities	Request compensation
3	Inform service center	Skipped activity	Service job under warranty
4	Inspect machine condition by mechanic and record service job information	Interchanged activities	Service job under warranty
5	Inspect machine condition by mechanic and record service job information	Interchanged activities	Inspect machine condition before service job
6	Issue spare parts	Different dependencies	Service job under warranty
7	Notify no payment to dealer	Different roles	Request compensation
8	Notify no payment to dealer	Different dependencies	Request compensation
9	Pay money to dealer	Different roles	Request compensation
10	Pay money to dealer	Different dependencies	Request compensation
11	Send to headquarters employee	Corresponding collections of activities	Request compensation
12	Send to headquarters employee	Different dependencies	Request compensation
13	Verify problem	Different dependencies	Service job under warranty
14	Verify service job (final) and customer signature in service job paper	Corresponding collections of activities	Request compensation
15	Verify service job (preliminary) and customer signature in service job paper	Corresponding collections of activities	Request compensation

Fig. 8. Example of change impact report with regard to a certain kind of artifact.

References

[1] Object Management Group. (2011, January 3). Business Process Model and Notation (BPMN) [Online]. Available: http://www.omg.org/spec/BPMN/2.0/PDF

[2] F. A. C. Pinheiro and J. A. Goguen, “An object-oriented tool for tracing requirements,” *IEEE Software*, March 1996, pp. 52-64.

[3] S. A. Bohner and R.S. Arnold, *Software Change Impact Analysis*. Los Alamitos, California, USA: IEEE Computer Society Press, 1996.

[4] B. Piprani, M. Borg, J. Chabot, and É. Chartrand, “An adaptable ORM metamodel to support traceability of business requirements across system development life cycle phases,” in *Proc. On the Move to Meaningful Internet Systems: OTM 2008 Workshops*, Lecture Notes in Computer Science Volume 5333, 2008, pp. 728-737.

[5] IBM. (2013, December 8). Mapping Business Process Diagrams to UML Use Case Diagrams [Online]. Available: http://pic.dhe.ibm.com/infocenter/rsysarch/v11/index.jsp?topic=/com.ibm.sa.bpr.doc/topics/t_ovwmapbp2uml.html

[6] Y. Wang, J. Yang, and W. Zhao, “Change impact analysis for service based business processes,” in *Proc. IEEE Int. Conf. Service-Oriented Computing and Applications (SOCA)*, Perth, WA, 2010, pp. 1-8.

[7] H. Xiao, J. Guo, and Y. Zou, “Supporting change impact analysis for service oriented business applications,” in *Proc. Int. Workshop Systems Development in SOA Environments (SDSOA)*, Minneapolis, MN, 2007, 6 pp.

[8] R. Dijkman, “A classification of differences between similar business processes,” in *Proc. 11th IEEE Int. Conf. Enterprise Distributed Object Computing Conference (EDOC)*, Annapolis, MD, 2007, pp. 37-47.