Maintenance of customized processes

Pavel Bulanov*
supervised by Marco Aiello*

{p.bulanov,m.aiello}@rug.nl
University of Groningen
Nijenborgh 9, 9747AG Groningen,
The Netherlands

Abstract. When similar business processes are deployed in different organizations, one typically adapts manually a template process to the different execution environments and organizational needs. Service-Oriented Architectures facilitate the task of adaptation by exposing the functionalities needed by the process in an abstract way, decoupled from the implementation. Nevertheless, manual intervention to customize the processes is still highly necessary, making it hard to maintain the processes obtained as the result of customizations. Even more complex is the situation where several processes are customized independently but they are still treated as linked with each other or inherited from one base (reference) process.
In this paper, we provide an initial investigation of customization of controllable processes.

Keywords: Variability management, Customization, eGovernment

1 Introduction

The idea of having one process to fit all needs is not realistic. In general, one either designs a process for a very specific task, or designs a template of a process more or less formally, to be adapted by different parties to their specific situation. Governmental laws are a good example of such circumstances. The central government approves a law, which is a collection of requirements expressed in a peculiar type of natural language. The law is then accompanied by an interpretation document which explains how to implement it, again expressed in natural language. Then the bodies involved in the law need to take action in order to manage/enforce/adapt to the law's indications. One example is the Dutch WMO law[1] that mandates, for instance, the rules for providing publicly subsidized wheel chairs to citizens by the municipalities.

* Distributed Systems Group
[1] Wet maatschappelijke ondersteuning, Social Support Act approved in 2007 in the Netherlands.

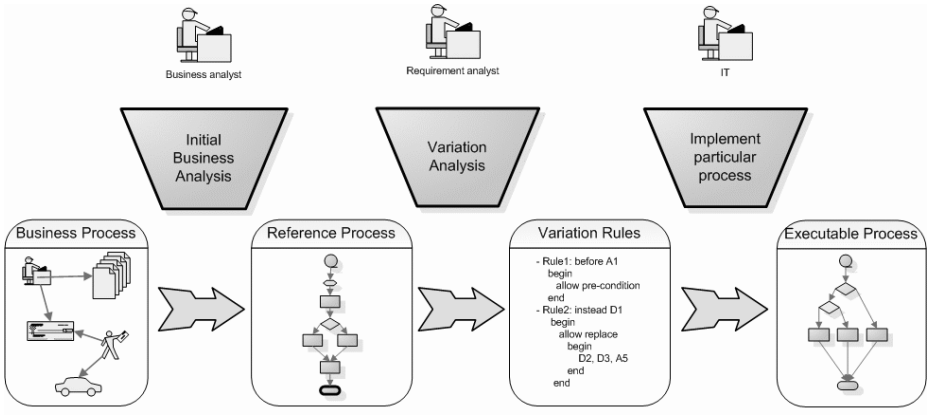

Fig. 1. Overview of the two-phase transformation.

The service-oriented approach, together with initiatives on corresponding standardization and implementation are strongly pursued by the Dutch government. In this work we deal with more technical aspects of these initiatives. We concentrate on the area of workflow management, which has seen great progress in the last years (e.g., [1]) and, if we consider the brand of Service-oriented architectures known as Web services [5], the natural candidate for modeling processes is the Business Process Execution Language (BPEL) [2].

However, BPEL has some limitations with respect to the vision of process customization. BPEL represents a business process as a tree-like structure with the individual activities as nodes, and the control of flow statements (if-else, while etc) as branching points within this tree. If the process itself is complex then this tree will also be complex and hard to understand by a human reader. Yet more problems rise while the process needs to be customized. There are several extensions like [3] or [4] which introduce a way to simplify representation of the complex business process via dividing it into the two parts — the "core" process and the additional rules over it. It simplifies the understanding of the whole process while providing a way to make customizations in the form of additional rules.

In case of a regulated environment while there is a need in keeping control over the possible customizations, those extensions will not help since they do not provide an explicit control over customization process. Such a control should be performed manually and there is no guarantee the changes applied are compatible with the existing regulation constraints.

To manage customization in a regulated environment, such as that of eGovernment, we propose a process language that allows us to express variations and constraints, and also a two phase transformation framework to manage the customization and maintenance of the processes. The overall transformation is illustrated in Figure 1. On the left, one starts with a description of a process, perhaps informal. The business analyst then transforms this into an unambigu-

ous process description and then adds variations and constraints to make sure that the process described is as widely applicable as possible. This concludes phase 1, the output of which is a process with variations described and a set of constraints to be satisfied. Phase 2 consists of the customization of the process at the many different points of execution. This last step is fully automatize and initiated by an IT specialist.

2 Issues with process maintenance

The proposed approach makes it also easier to maintain process evolution. For example, consider a situation when a reference process is changed due to an error fix or a change in the requirements. In this case it is required to repeat the second phase of transformation in order to propagate the changes into all inherited process automatically. Such advantage is achieved via the restricting possible customization as well as splitting the customization into the two formal phases.

On the other hand, it is impossible to anticipate all the possible changes therefore the proposed solution does not work for all cases. This is why the solution is offered for regulated environments such as eGovernment where it is easier to restrict the possible changes by means of administrative rules. Also under some circumstances it is needed to change the inherited process directly due to local needs which are not predicted by the original design. The whole picture is illustrated in Figure 2. Here the reference and inherited processes are displayed and the horizontal arrow labeled with "1" represents the existing transformation rules which are forming the link between the reference and inherited processes.

The reference process may also be changed leading to the new reference process, this is illustrated by the left vertical arrow labeled with "2". After than having the link between the reference and inherited process it is possible to build the new inherited process via repeating the underlying transformation. Obviously this transformation may become invalid over the changed reference process and thus the first issue appears: *How to identify are the changes being applied on the reference process compatible with all inherited processes?*

At the same time the inherited process may be changed which is reflected by the right vertical arrow labeled with "4", and here is the second issue: *Once the inherited process has changed, is it feasible to keep the link with the reference process?*

Note that the transformations "1" and "3" are the same since they both represent the link between the reference and inherited process. The problem is the transformation "3" may be invalid, as already described as the issue one.

3 Concluding remarks

A several approaches may be identified in order to solve aforementioned issues:

- Make an automatic merge of two workflows basing on formal graph transformation rules

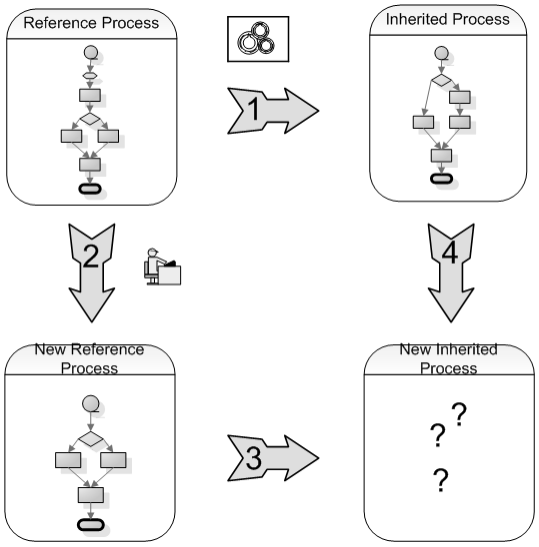

Fig. 2. The process maintenance.

- Confine the possible customization changes so it would be easy to perform a maintenance — the "difficult" cases in customization just not allowed
- Identify a "change set" or "change region" and treat it like a base building block of customization.

The first approach requires complex analysis of the merged graph since apart from the geometrical difference there is also semantic meaning of each node as well as the side effects. This way itself stands a dedicated challenge and therefore is beyond the scope of this paper.

The second approach is the easiest one but loses the flexibility thus making it difficult to use for real world tasks. This approach as based on assumption the particular "difficult" customization cases are not important and may be suppressed. Such assumption is valid for dedicated domains only and therefore is not suitable for a general–purpose solution.

For the third approach there are many investigations related to the identifying of a change set, for example [6]. The extraction of a change region may help to construct a conversion function as an interpretation of manual changes. This conversion function corresponds either the arrow "4" or the arrow "2" on Figure 2, depending on which process is changed. Once the conversion function is extracted it is possible to use the combination with the existing transformation function — either the combination of "2" and "3" (when the reference process was changed) or "1" and "4" (when the inherited process was changed).

The main issue there is to extract the conversion function correctly and to identify possible cross-influence between this extracted conversion function and the existing transformation link, which is the subject for future research.

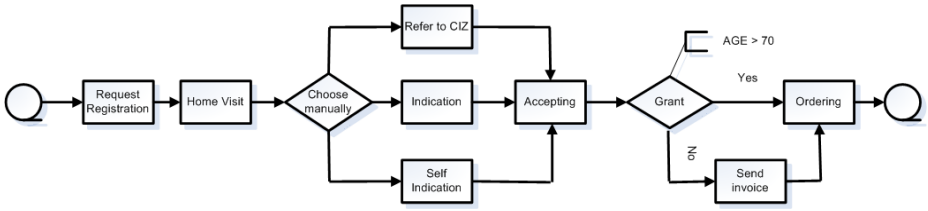

Fig. 3. An example of an e-Government process for obtaining a wheelchair.

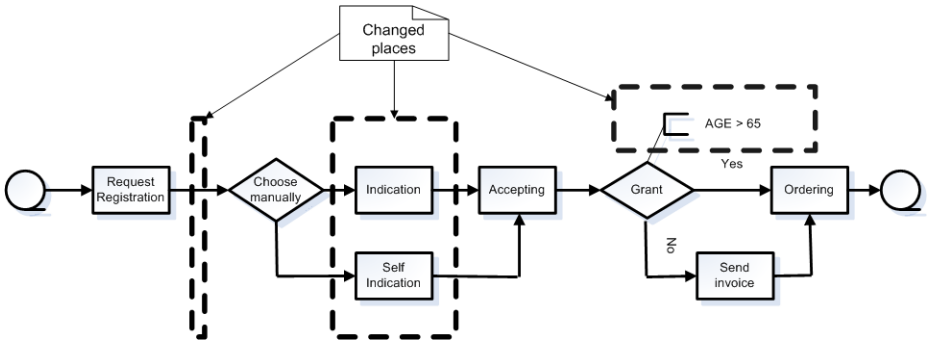

Fig. 4. Example 1

Consider an example of a typical eGovernment process — issuing a wheelchair for elderly people (see Figure 3). Reading the figure from left to right, we have the initial activity of registering for obtaining a subsidized wheelchair. The next activity is that of having an expert from the municipality visiting the home of the requesting actor. Then there is a choice point requiring a choice being made by an authorized civil servant. Based on this choice three further activities are possible, which then rejoin in an accepting the request activity. Now a further automatic choice is performed, such as a check on the age of the requesting party. At the end, one will receive a wheelchair having to pay for it himself, or having it subsidized. Furthermore, in one municipality the minimum age for getting a wheelchair may be 70, and in another one it may depend not only on the age.

Now, consider the case where the reference process is not acceptable for one of the local municipalities, so they made an appropriate change and end up with a process shown on Figure 4. The changed regions are outlined with dotted rectangles.

What happens if the reference process itself is changed after the customization has been done? For example, the reference process is modified in a way when the activity "Refer to CIZ" is not needed. That means the change is not dangerous since this activity is not used in the inherited process as well but on the other hand this activity belongs to the *change region* and therefore it may lead difficulties with automatic resolving.

Another example the reference process change is the removing of the activity "Indication" and this activity belongs to the *change region* and also used by the inherited process. Two options are available — remove this activity from the inherited process as well or report an error since such change is illegal. It is not possible to identify which option is correct since it depends on the logic beyond processes and changes made.

Those simple examples show the difficulties with the automatic process maintenance therefore making the task of automatic process maintenance challenging.

Acknowledgements

The research presented in this paper is supported by the NWO Jacquard Software Engineering program with the project Software as a Service for the varying needs of Local eGovernment (**SaS-LeG**), with partners: the University of Groningen, Cordys and an association of Dutch Municipalities. Contract no. 638.000.000.07N07. Project website: http://www.sas-leg.net.

References

1. W. van der Aalst and K. van Hee. *Workflow Management.* MIT Press, 2002.
2. BPEL4WS. *Business Process Execution Language for Web Services*, May 2003.
3. Anis Charfi and Mira Mezini. Hybrid web service composition: business processes meet business rules. In *ICSOC*, pages 30–38, 2004.
4. M. Momotko, E. Tambouris, G. Blizniuk, W. Izdebski, and K. Tarabanis. Towards implementation of life-events using generic workflows. In *eGovernment Workshop '06 (eGOV06), Hosted at Brunel University*, 2006.
5. Mike P. Papazoglou and Willem-Jan van den Heuvel. Service oriented architectures: approaches, technologies and research issues. *VLDB J.*, 16(3):389–415, 2007.
6. Wil M. P. van der Aalst. Exterminating the dynamic change bug: A concrete approach to support workflow change. *Information Systems Frontiers*, 3(3):297–317, 2001.