

# Towards Creating Workflows On-the-Fly and Maintaining Them Using the Semantic Web: The sButler Project at Linköpings universitet

Nahid Shahmehri

Dept. of Computer and Info. Science  
Linköpings universitet  
SE-581 83 Linköping, Sweden  
+46 13 28 20 66

nahsh@ida.liu.se

Juha Takkinen

Dept. of Computer and Info. Science  
Linköpings universitet  
SE-581 83 Linköping, Sweden  
+46 13 28 26 03

juhta@ida.liu.se

Cécile Åberg

Dept. of Computer and Info. Science  
Linköpings universitet  
SE-581 83 Linköping, Sweden  
+46 13 28 89 86

cecab@ida.liu.se

## ABSTRACT

In the sButler project (Semantic Web Butler) at Linköpings universitet, Sweden, we examine the modeling and instantiation of workflows based on Web services (services described in a rule-based language). The aim of the project, which is part of the Swedish Semantic Web initiative [1], is to support on-the fly creation of workflows and their subsequent maintenance. Our sButler agent will manage workflows and Web services modeled as Petri Nets, use the language facilities provided by the Semantic Web, and employ a new push-and-pull protocol to handle communication with other sButlers.

## Categories and Subject Descriptors

D.2.13 [Software Engineering]: Reusable Software - *domain engineering and reuse models*. H.4.1 [Information Systems Applications]: Office Automation - *workflow management*. D.2.2 [Software Engineering]: Design Tools and Techniques - *Petri nets*. C.2.2 [Computer-Communication Networks]: Network Protocols - *applications*.

## General Terms

Design, Languages, Theory

## Keywords

Semantic Web, Web services, agents, workflow, ontology.

## 1. INTRODUCTION

This project concerns workflows that use Web services. Such workflows are executed in a Semantic Web environment and must cope with some practical characteristics of Web services. For example, the Web services are not always available, and they can sporadically be replaced by new ones. Also, the Web services will not always match the complex needs of a user, so *composition* [2][3] is necessary, in addition to locating and comparing Web services.

Given a specialised workflow described in a composition language, the sButler agent will allow the user to delegate some part of the workflow for external execution as a Web service. Moreover, whenever a Web service that is part of a workflow appears, disappears, or changes its interface to the Web, the sButler will react to ensure that the workflow is always correctly instantiated.

We are examining the modelling of the agent behaviour in this context, and also the networking protocol needed for communication between agents and between agents and services/workflows. Our goal is to develop a practical tool. For this purpose we are collaborating with the industry [8], and also striving for the use of standard

protocols and frameworks as much as possible.

## 1.1 The sButler Functions

The sButler functions are useful in many different contexts. The sButler helps the user to manage her workflow. It can suggest changes to the workflow (“You need to update your EDI”) and even perform the change for the user (“Do you want me to update it?”).

The sButler helps the user to keep contact with some specific Web services. For example, the user has a contract with a specific provider of office supplies and needs support to deal with the possible workflow adjustments that this implies. On request from the user (her personalised and specific needs), the sButler searches for appropriate Web Services (“fastest”, “most reliable”, “cheapest”, etc., according to the user’s preferences) to plug in to the workflow of the user. Parts of the user’s workflow can be flexible with regard to who owns the Web service that is to be included. For example, if the Web service changed the credit card from Master Card to Visa, the user can accept to change its workflow’s input (the credit card) to fit this change.

The sButler maintains a “best instantiation” of the delegated part of the workflow by constantly checking the Web services available according to the user’s requirements. For example, when ordering new supplies for the office, the sButler can ensure that each order will be sent to the provider that currently offers the best price for these goods. To summarise, the sButler can:

- Locate Web service providers.
- Maintain the composition of Web services as part of the user’s workflow. When a Web service is no longer available, or changes its interface, the sButler is responsible for repairing it.
- Keep the user’s workflow up-to-date by suggesting changes matching requirements of the user and the workflow.
- Support the creation of a new workflow.

## 1.2 Project Components

Figure 1 below illustrates the usage of the sButler: The user wants to delegate a task from her workflow to an external Web service. She invokes the sButler via a Web browser and specifies the task and additional requirements that she has on the Web service(s). The sButler initiates its service discovery function and returns the “best” matching service to the user. The sButler then establishes a connection to the service provider’s sButler so as to be notified of any future changes in the service profile. The Semantic Web capabilities (and domain-specific ontologies developed for the sButler) are employed in the description of the user’s requirements and the task description, and communication with other sButlers and service providers. If a service disappears, the sButler is notified and restarts its service discovery. It can also receive messages from other sButlers

about new services, which can be compared to existing ones and also possibly replace them (if “better” match).

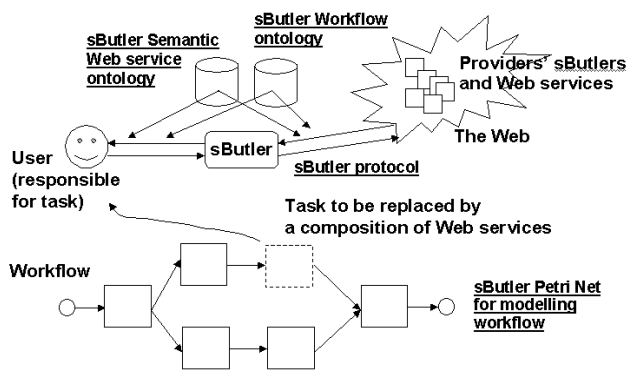


Figure 1 The sButler components.

We need the following sButler components (underlined in figure):

- A modelling language for describing workflows and Web services. We are currently surveying several approaches and leaning towards the use of Petri Nets, a powerful tool for workflow specification and simulation [1][7]. The approach must include the capability for both workflow description and network protocol specification since Web services are components with protocols, or processes with workflows to be composed.
- A network push-and-pull protocol for communication between sButlers in the application layer. This is used by the sButler to push information about Web service changes, and to pull updates of Web services as needed, with a minimum of network traffic. We are examining the Tacoma push-protocol [12][10], which we plan to extend into a push-and-pull protocol.
- Ontologies [5] for describing the user's requirements and changes in workflows and Web services. Our approach for building ontologies is described in the next section.

## 2. ONTOLOGIES FOR sBUTLER

We have identified two ontologies needed in sButler: one for describing workflows, and one for changes in Web services.

We have decided to use the DAML+OIL [4] framework to implement the ontologies because it has a rich development environment. DAML+OIL is also close to a Web standard for Semantic Web services. To our knowledge there is no previous DAML+OIL ontology for describing a workflow. We consider DAML-S [4] concepts of process composition to be expressive enough for describing simple workflow structures. We use it as a prototype language to understand which concepts are important for describing user requirements (modelled as workflows) in sButler.

We have defined a first version of an ontology for changes in Web services. Figure 2 illustrates its use. The ontology refers to Web services that have been described with an existing Web service ontology. The Web service ontology in the figure is DAML-S. One of the preconditions of the Web service specifying the category of the credit card used to pay for the service has suddenly changed from Master Card to Visa, thus affecting the functionality of the service.

The example describes the change in one of the service *properties* (DAML-S profiles consist of a number of input, output, precondition and effect properties). Other types of changes can occur with regard to a service. Our ontology can also describe changes to the *structure* of the service (in DAML-S terms, the

change affects the process composition description), and changes affecting the *whole service* (appearing/disappearing).

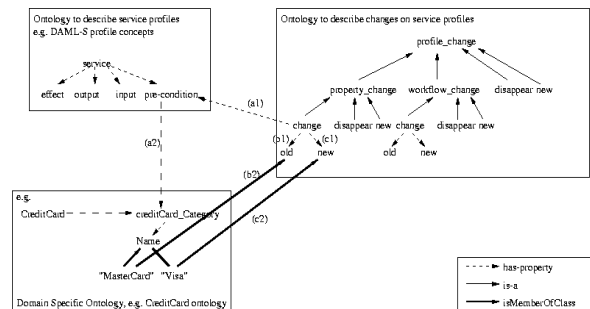


Figure 2 An ontology for service changes in sButler.

## 3. CONCLUSION AND FUTURE WORK

We have described the ongoing sButler project at Linköpings universitet, aimed at supporting and maintaining the relation between the workflows of companies/users and existing Web services/providers. The project is in a starting phase and includes the development of a base ontology for Web service workflows ontologies, a Petri Net-based Web service workflow specification techniques, and a Web service push-and-pull protocol. We are currently implementing a prototype based on a standard platform [9], and also analysing the design space of protocol specifications and workflow descriptions. The project lasts until summer 2005.

## 4. ACKNOWLEDGMENTS

We thank Vinnova [13] for supporting this project.

## 5. REFERENCES

- [1] van der Aalst, W. M. P. (1998), “The Application of Petri Nets to Workflow Management”, in *The Journal of Circuits, Systems and Computers*, pp. 21-66, Vol. 8, No. 1, 1998.
- [2] Assmann, U. (2003), *Invasive Software Composition*. Springer.
- [3] Cardoso, J. & Sheth, A. (2002), *Semantic e-Workflow Composition*. Technical Report, LSDIS Lab, Computer Science Dept., Univ. of Georgia, USA.
- [4] – (2003), *DAML*. Web. <http://www.daml.org/>
- [5] Fensel, D. (2001), *Ontologies: A Silver Bullet for Knowledge Representation and Electronic Commerce*. Springer.
- [6] Hedström, N. (2003), *Design, implementation och utvärdering av en workflowmotor*. Master thesis. LiTH-IDA-Ex-03/12, Dept. of Computer and Info. Sc., Linköpings universitet, Sweden.
- [7] Jensen, K. (1987), “Coloured Petri Nets”, in *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986 Part I, Lecture Notes in Comp. Sc. Vol. 254*, pp. 248-299. Springer.
- [8] – (2003), *Medius Ibitech*. Web. <http://www.medius.se/>
- [9] – (2003), *Microsoft.NET*. Web. <http://www.microsoft.com/net/>
- [10] Sudmann, N. P. & Johansen, D. (2002), “Software Deployment Using Mobile Agents” in *Proc. of First Int'l IFIP/ACM Working Conf. on Component Deployment, June 20-21, 2002, Berlin*.
- [11] – (2003), *SWEB Portal*. Web. <http://www.ida.liu.se/zope/portals/sweb>
- [12] – (2003), *TACOMA: Operating system support for agents*. Web. <http://tacoma.cs.uit.no/>
- [13] – (2003), *Vinnova*. Web. <http://www.vinnova.se/>