

TOWARDS EFFICIENT AND TRANSPARENT E-GOVERNMENT PROCESSES

Abdelbaset Rabaiah

*ETRO Research Group, Vrije Universiteit Brussel
1050 Pleinlaan, 2
Brussels, Belgium
aabdelgh@vub.ac.be*

Eddy Vandijck, Professor

*ETRO Research Group, Vrije Universiteit Brussel
1050 Pleinlaan, 2
Brussels, Belgium
Eddy.Vandijck@vub.ac.be*

ABSTRACT

Service-Oriented Architecture (SOA) is gaining momentum as an architectural model for heterogeneous e-Government systems. With SOA, e-Government is perceived as a collection of Web services. A real-life event (e.g. a newborn) triggers a sequence of a number of services. This sequence is usually part of a process. A process has input requirements and produces a final output (e.g. a printed birth certificate). For a government there is a myriad of processes. Not all services are electronic though. This depends on the maturity of the e-Government endeavour. For all processes, the sequence of services must be fed somehow into the system. This enables the system to discern what services to invoke and what input requirements are needed for each process. Electronic services can be invoked instantly. Traditional ones are flagged to be carried out manually. Processes take the form of if-then scenarios. This paper proposes the use of a rule-based approach to implement processes. As we shall see, this approach eases the maintainability of the plethora of e-Government processes. It does not incur recoding of applications should there be a change to one or more processes. A process sequence is not uniform for the same type of event. Two citizens/customers having the same real-life event can go through different sequences of a process to get the same output. Our rule-based approach can inherently automatically inform the citizen what input requirement is missing and why. It can also inform the citizen why a process sequence is particularly so for his/her case. This raises transparency to a great extent. Our approach is intuitive and natural. It makes it easier for public servants themselves lacking traditional programming skills to update the rules. To validate our approach we have built a prototypical implementation to prove workability.

KEYWORDS

E-Government, Process, Rule-based, SOA, Logic Programming.

1. INTRODUCTION

Governments worldwide are accelerating their efforts to offer online services. Most have launched different projects to move forward towards full electronic government (e-Government) implementation. One of the biggest challenges of e-Government is interoperability. Most government agencies have back office systems installed. Development of such systems used to be carried out separately to meet local needs. Decision making in government agencies is decentralised. This has resulted in incompatible back office systems. Application programs, operating systems data formats and hardware characteristics have little in common. (Rabaiah et al. 2006) proposed a model that insures interoperability among disparate systems. Federation of e-Government which is based on SOA was argued to advantageous in the sense that it simplifies e-Government implementation and maintenance.

Decentralisation and lack of system interoperability have had their influences on government processes. Klischewski (2001) compares the service flow (which is now the articulated SOA) and workflow approaches.

Service flow better fits governmental processes as there is no central flow "engine" to coordinate order of execution of the different services offered by different government agencies. Processes in governments are decentralised. In other words, processes follow a choreographic paradigm. Process ownership changes continuously across the different agencies involved.

SOA allows for citizen-centric processes. According to (Klischewski 2001) service flows constitute personalised sequence of interrelated activities. A process in the service flow model is considered as a means for personalisation rather than control.

In many cases, application development mixes up process logic, application logic, GUI logic and data access logic (Tammet et al. 2006). The obvious downside in the context of this paper is that governmental processes become less understandable. This has its ramifications on the maintenance, reusability, and portability of the process logic. It is the purpose of this paper to propose a system that can keep the logic of governmental processes separate and clearly-defined at all times. To achieve this objective we promote the use of a rule-based methodology in designing e-Government processes.

2. WHY USE RULE-BASED METHODOLOGY FOR E-GOVERNMENT PROCESSES?

The role of a government is to practice governance and maintain the well-being of the society. In contrast the role of business is to achieve maximum profit with least cost. This huge difference in role has its influence on processes. For a government process, a number of strict considerations have to be taken into account at design time and at run time.

Government processes follow delicate legal constraints. Legal constraints come first in the planning of government processes. After all, government is supposed to enforce law and order. Failure to do so ushers corruption and even chaos. Corruption in a government has far more impact on the population than that in a business. Governments -especially democratic ones- make sure that they abide by the law. One way to manifest this commitment is to expose their processes. Thus, transparency is an issue in government processes. Corruption and lack of transparency has severe political and social consequences. With this ugly formula there is no way to address accountability. Furthermore, persecution and bias can become so widespread.

A government process is a logic-intensive knowledge that is too intricate to be represented with traditional procedural programming or databases. Think about taxation for example. Traditional procedural programming tools are omni-available. In addition, skilled programmers who can use these tools, outnumber logic programmers considerably. These tools, however, fall short in representing logic knowledge. Logic rules exhibit non-procedural sequences of execution.

Transparency in government processes, per se, is a constituency or public demand. Normally, there exists some independent organisations that conduct auditing of government operation including processes. In many cases, these auditors do not possess the appropriate IT background. In this case, separate and simple rule-based representation of processes will come in handy. Otherwise, auditing becomes ineffective.

Business processes' design, on the other hand, focuses on efficiency, cost reduction, and capitalization on profit. Business processes generally produce products. Government processes, by contrast, aim to furnish citizens and businesses with services (e.g. issuing a permission to build a house or to start a new business). Clearly, government processes, in general, are more delicate and therefore should be represented accurately.

It is well-known that logic languages are much more expressive than procedural languages. Non logic-based programming languages use basically variables to represent facts. The extensive use of variables in complex structures (e.g. loops, branching...etc) causes a degradation in the readability of the program which renders it harder to maintain or update. Whereas, logic languages provide a simpler and a declarative representation of facts and rules in a coherent manner and then use either forward chaining or backward chaining to execute the programme. Logic programs are also pointer-free which makes them much more flexible than traditional programs (Morozov et al. 1999).

Furthermore, a simple declarative representation of facts and rules makes it easier to be understood by humans. This representation has a self-describing nature. It makes it as a result "self-documenting". This is necessary on the long run. It is not very uncommon that software manuals get lost after a long period of time.

In cases like these it would be almost impossible to update the software without rewriting it. It becomes worse if the contract with the software developer has long ended.

In many cases, rules have time-frame validity. A specific rule or a collection of rules can be valid for a certain period of time. Other bundle of rules might become applicable afterwards. This means that different groups of rules are applied across the time-line of a process. In such cases, there is a need to keep track of which rules were applicable at which time-frame. One of the persistent problems of using traditional databases to store rules was the difficulty to express time-dependant rules. With the rule-based approach it is much simpler and easier. Time validity is embedded within the rules themselves. All rules are kept in one separate place called “Knowledge Base”.

Prolog as a logic language is very much capable of natural language recognition. It allows clients to get recommendations on the best things to do in order to get a government service. The client can feed information to the system in natural language. The system then analyses what has been typed, recognises it, and responds accordingly by choosing the best process. Finding the best track or process for the client to go through is one aspect at which logic programs excel. Logic languages are much more advanced in searching best solutions than traditional procedural languages (Gazdar & Mellish 1989).

2.1 Are Changes to Process Logic in Governments Common?

One thing for sure is that process re-engineering and streamlining is much more evident in businesses than in governments. This has to do with the fact that business environment is ever changing due to microeconomic factors and fierce competition. Still though, changes to government processes are more frequent than might seem to be. It seems that government processes are frequently reviewed. Reasons behind process re-design vary from policy shifting to enhancing transparency and efficiency.

There are many examples that took place in different countries. To give just one significant example we will introduce the Belgian initiative “Kafka”. In 2003 the Belgian Secretary for Administrative Reform was assigned a specific task: cut the red tape (Kafka 2003). This initiative was meant to commence large-scale administrative reforms that will rid citizens and businesses of administrative rigmarole. Notorious bureaucracy caused great inefficiencies for citizens and businesses alike. In just four years some (200) laws and regulations were either simplified or even abolished all together (Kafka Report 2006). As a result administrative costs for businesses have decreased by 25%, a reduction of about (1.7) billion euros. This has put Belgium in first position among European countries where it is fastest to start a new business. According to the World Bank, “Belgium has developed the European Union’s most innovative communication program for cutting red tape”.

From the description of the case above we realise that changes to government processes can occur more frequently than one can imagine. This requires that process implementation should be designed in such a way that allows updates and maintenance easily.

3. THE PROPOSED APPROACH

As mentioned earlier, there is a need to keep the logic of governmental processes separate, clearly-defined and reusable. To achieve this, some assumptions must be made regarding the proposed approach:

- The proposed approach should be logic-based as processes themselves are logical (rule-based)
- Rules should be kept separate from procedures (application programs) and not embedded in them
- Rules should be declarative, none procedural and intuitive
- Rules are understood/written by people who presumably do not have programming skill
- Rule-base development should take the least amount of time
- It should be easy to maintain or update the rules
- Rules should be reusable (portable)
- The approach should allow explanation and reasoning to provide the sought level of transparency to “customers”

Once a system possesses these qualities it merits as being the answer for what we need to achieve. What is needed now is to compile the tools necessary to originate such a system. The following section describes

our rule-based prototype. It has proven the workability of the rule based approach for e-Government processes in an SOA environment.

4. OUR PROTOTYPE

Until recently, the most prominent logic programming language Prolog suffered from two major deficiencies to be used for Web development: embeddability and extensibility. Embeddability means the ability to embed Prolog code in other programming languages. In other words, it means: ability to provide APIs in other languages for Prolog programs. Extensibility, on the other hand, refers to customizing Prolog programs by extending them with other programming languages (e.g. Java). These two shortcomings were overcome with the creation of tools that bridge Prolog code with code of other procedural languages like Java. An example of such tools is Amzi!Prolog which we used in our prototype.

We chose Amzi!Prolog (Amzi, 2006) as our Logic Server because it was free for academic use. This Prolog Engine supports the embedment of Prolog code into host languages like C++, Java...etc. We chose Java as our host language because of its widespread use in Web applications. We compiled our sample Prolog program which serves as our mini Knowledge Base. A knowledge base is essentially a collection of facts and rules written in Prolog (see Figure 1). We then used the provided Amzi!Prolog Linker to generate the loadable file.

After that, we utilised the readily available Java APIs to connect to the mini Knowledge Base. We followed the Amzi!Prolog guidelines to do that. Then we installed Tomcat as our Web Application Container (Web server). Here, we wrote our sample Web application. Finally, we installed the Web Service extension facility. AXIS was used on top of Tomcat server (see Figure 1). At the end, we wrote the WSDL document to connect to the applications Web Server. With this we had a fully-functional Logic Web Service running. We found that rule-based process development was much more efficient than traditional procedural counterpart in the context of e-Government processes. We could alter rules very easily and quickly without hard-coding. Maintaining and updating the rules was easily attainable by editing the separate file(s) of facts and rules. This is something impossible with traditional procedural programming. Execution time, however, was slightly longer for simple processes compared with procedural programming but for complex ones it was more efficient. For the objectives of this paper, our prototype has proven the workability of the rule-based approach.

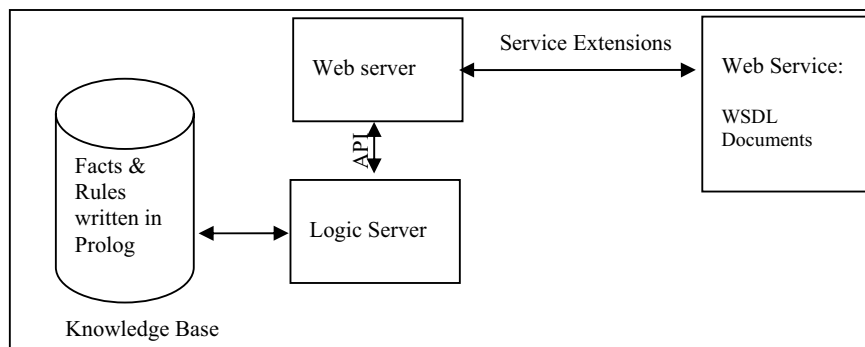


Figure 1. Prototype architecture

Prolog has the capacity to be used for natural language recognition (Gazdar & Mellish, 1989). A citizen for example, can express his query in his own language over the Web and get accurate recommendation. By doing that, we can create an expert system that is available twenty four hours a day and seven days a week. This enhances government service dramatically. Moreover, the system can read the rules and facts in the knowledge base and easily generates a report about procedures, conditions, requirements...etc of a particular government service. The information in such a report will be up-to-date. The citizen or company can then know why the procedure is as such. This enhances transparency.

We produced an intra-WS prototype that implemented processes within a WS as rules and facts. Still though, we did not resolve the inter-WS rule-based implementation of processes. What is needed is an XML together with a rule-based approach. XML is needed for portability. A choreographic language (e.g. WSCI

which is XML-based) is needed but with ability to support logic rules. These rules can either be natural language-like or formal notation or even a mixture of both. We believe that this should be a direction for new research for better process implementation. The authors are joining a group of researchers from both academia and industry to come up with such a language. An initiative called "The Rule Markup Language (RuleML) Initiative" is being launched (www.ruleml.org). It is about bringing logic and XML together.

5. CONCLUSION

We have presented a rule-based approach for implementing e-Government processes. In our prototype we have managed to keep the logic of processes separate and well-expressed. It is a promising approach to reduce complexities in Service-Oriented Architecture which has received interest for e-Government implementation. Typically, SOA supports a multitude of electronic processes of a government or a business. Part of these processes span across more than one Web Service. Our approach helps sort out the logic behind these processes so that it remains understandable and auditable.

The proposed approach bridged eloquent logic declarative programming with the power of procedural programming over the Web. This *modus operandi* has the best of both worlds. On one hand, we use the precise description of facts and rules of processes. On the other hand, we use the efficient and widely used procedural programming. In such methodology, we split-up logic and operation. This has a unique advantage for simplifying understanding and maintenance of e-Government processes.

REFERENCES

- de Carvalho, C. L., 2001. NetProlog Home Page. Website: <http://netprolog.pdc.dk>, last visited: August 25, 2006.
- Friedman-Hill E. J., 1999. *JESS: The Java Expert System Shell (Version 5.0, alpha)*, Website: <http://herzberg.ca.sandia.gov/jess>, last visited: August 27, 2006.
- Gazdar, G., Mellish, 1989. *Natural Processing in PROLOG: An Introduction to computational Linguistics*, Addison Wesley. 0 201 18053 7.
- Gregory Karvounarakis et al, 2002. RQL: a declarative query language for RDF. *Proceedings of the 11th international conference on World Wide Web*, Pp. 592-603. ISBN:1-58113-449-5.
- Jenny, Y.Y. et al, 2007. Effective e-Government Process Monitoring and Interoperation- A Case Study on the Removal of Unauthorized Building Works in Hong Kong. *Proceedings of the 40th Hawaii International Conference on System Sciences*. 1530-1605/07, IEEE.
- Jim A. Cornwell, 1998. *History of Philosophy Ancient times Sixth to third century BC, The Alpha and the Omega*, Volume III.
- Kafka Report 2006. Available at <http://www.kafka.be>. Last visited March 1, 2007.
- Matjaz Juric, 2005. *BPEL And Java*. Available at: <http://www.inf.ed.ac.uk/teaching/courses/ec/miniatures/bpel-up.pdf>, Vited:April 29, 2007.
- Morozov, A. et al, 1999. On the Problem of Using Logic Object-Oriented Programming in the World Wide Web. *Proceedings of the Special Russian Session "The Internet Developments in Russia"*. First IEEE/Popov Workshop on Internet Technologies and Services, pp. 54-59.
- Rabaiah, A. et al, 2006. Abstraction of eGovernment. *Proceedings of the IADIS International Conference E-Commerce*. Barcelona, Spain, pp. 27-34, ISBN: 972-8924-23-2.
- Ralf Klischewski. (2001) Infrastructure for an e-Government Process Portal. *Proceedings of the European Conference*. UK, 2001, pp 233-245.
- Rule Markup Language, <http://www.ruleml.org/>. Last visited: February 13, 2007.
- Szeredi, P. et al, 1996. Serving Multiple HTML Clients from a Prolog Application. *Proceedings of the 1st Workshop on Logic Programming Tools for Internet Applications*. Bonn, Germany, pp. 81-90.
- Tarau, P., 1999. *Java Inference Engine and Networked Interdictor: A Prolog Interpreter in Java for Mobile Agent Scripting and Internet Programming*. MIT Press.
- Tammet, T. et al, 2006. A Rule-Based Approach to Web-Based Application Development. *IEEE 1-4244-0345-6/06*. Win-Prolog. - LPA. Website: <http://www.lpa.co.uk>. Last visited: Aug. 27, 2006.