An interoperable architecture and principles for implementing strategy and policy in operational processes

Yiwei Gong *, Marijn Janssen

Faculty of Technology, Policy and Management, Delft University of Technology, Jaffalaan 5, 2628 BX Delft, The Netherlands

Article history:
Received 6 April 2013
Accepted 25 June 2013
Available online 23 July 2013

Keywords:
Interoperability
Agility
System architecture
Architectural principles
Business process
Strategy implementation

ABSTRACT

In today’s economy managers expect new strategies and policies to be implemented quickly. Yet practice shows that current systems are not able to implement changes within a short time frame. Nowadays a variety of technologies including semantic web services, business rules and software agents are available as building blocks for interoperable systems. Yet these technologies are rarely used in combination and are not adapting operational processes to changing business strategies and other requirements. A generic architecture is presented in this paper which is able to adapt processes to changing requirements based on three types of knowledge repositories (domain ontology, service description and business rule) which, when combined, allows for direct employment of strategy in business processes. The architecture is tested using scenarios. The testing shows that the architecture improves the interoperability between policy/strategic and operational level and results in a higher agility and better compliance.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

Strategies and policies are influenced by all kind of developments and frequently change. Managers, customers and several other stakeholders expect changes in strategies and policies to be implemented quickly. Strategy formulation is often viewed as a conveyor belt in which opportunities and issues are recognized as problems, alternative courses of actions are formulated, and these are affected, implemented, executed and evaluated [1]. Armistead et al. [2] distinguished between operational or horizontal processes and direction-setting or vertical process (see Fig. 1). Horizontal processes are concerned with the production and delivery of products or services, while vertical processes involve strategy and policy formulation and deployment. Vertical processes are initiated by factors such as changes in customer needs, technology developments, market conditions and changes in the regulatory environment and often need to be implemented within a short time frame for reasons like competitiveness, legal compliance and dealing with changed circumstances. A lack of interoperability typically happens in the intertwinedness between the vertical and horizontal processes. Both processes are often developed and executed independently, and are typically supported by different systems. The implementation of strategies and policies cannot and should not be blocked or hindered by a lack of interoperable systems, but this is often the case as shown by some examples such as the Walvis [3]. In this situation a new policy was made for reducing the administrative burden by simplifying the employment insurance law, but it failed in implementation in time. A main issue in this example is the need to create interoperability between the strategic level and the administrative processes executing the policy, as this facilitates fast and low-cost implementation of new policies. Interoperability is the ability to exchange information between systems [4] and is necessary for connecting the systems supporting vertical and horizontal processes. Interoperability enables strategic and operational systems to exchange information with each other (Fig. 1). Only if these are connected the horizontal processes can adapt to the changes managed and stored in the vertical systems. Our system architecture will relate systems supporting strategy and operational process using business rules models.

Interoperability should enable agility, which can be defined as the speed in responding to variety and changes [5]. Being able to respond quickly is vital in a continuously changing environment in which the systems need to be continually improved in order to reflect changes [6]. However, whereas much literature is focused on horizontal processes, scant attention has been given to the relationship between vertical and horizontal processes. The focus of this research is on the adaptation of horizontal processes to strategy and policy changes. A generic architecture and principles is presented in this paper as the solution for adapting horizontal processes to changes from vertical processes.
In this paper a system architecture is proposed which allows agile implementation of new strategies and policies into operational processes. Along with the architecture, architectural principles that lay the foundation for the system architecture are provided. The rest of this article is organized as follows. In the next section the background and related work in the field of interoperability and architecture is presented. Next the design science research approach is presented followed by a description of the architecture. Then we present the architecture including the architectural principles. Thereafter architecture is illustrated by a prototype. The paper ends with a conclusion and prospects for future work.

2. Background and related work

Among many definitions of interoperability, the one provided by IEEE is most commonly used [7]: “the ability of two or more systems or components to exchange information and to use the information that has been exchanged” [4] (p. 114). Accordingly, interoperability can be between systems or within a system consisting of subsystems. In the context of strategy and policy implementation, interoperability can be viewed as the ability to translate strategy and policy that are stored in systems into software systems that support operational processes. It concerns the use of ICTs to facilitate the coordination of work and information flows [8]. Such ability is reflected by the agility of strategy and policy implementation, namely the speed, costs and the quality (e.g. compliance) of strategy and policy implementation [3]. In the evaluation of the architecture we will look therefore at these four evaluation criteria:

1. Agility – the speed of implementing new strategy and policy [9];
2. System flexibility – the ability to adjust processes and include new subsystems or components [10];
3. Compliance – meeting the requirement imposed by strategies, policies and regulations on the tasks that are performed by organizations [3]. In essence compliance is about whether strategies, policies and regulations are reflected in operational processes;

According to Naudet et al. [7], the need for interoperability arises when two or more incompatible systems are placed in relation. Interoperability is often studied from a technical perspective (cf. [12]), where different ICT applications are considered to be individual technical systems that require interoperability to connect with each other. Interoperability is often studied from a semantic perspective (cf. [8]), where the information exchanged between heterogeneous systems should be meaningful and all the communicating parts should interpret it in the same way. Interoperability has been studied from an organizational perspective (cf. [13]), where different departments are thought of as individual sub-organizational systems that require interoperability to cooperate with each other.

In literature several approaches for achieving interoperability have been developed including, standards (e.g. [14]), reference models (e.g. [15]), architectures (e.g. [16]), and frameworks (e.g. [17]) [8]. However, strategy and policy implementation rarely receives attention. General interoperability frameworks like the European Interoperability Framework [17] and INTEROP NOE [15] framework mainly help to identify and structure the knowledge of the domain e.g. “interoperability concerns” (data, services, processes and business) and “interoperability barriers” (conceptual, organizational and technological). Although these frameworks can be used to classify and structure the basic aspects of interoperability, their intent is not to provide working solutions for the implementation of strategies and policies [17]. In strategy implementation interoperability is necessary to translate strategies and policies into executable business processes. In such a situation vertical and horizontal processes are intertwined, as the outcome of vertical processes influences the horizontal processes. This intertwining is the focus of this research. Interoperability frameworks provide structure of concepts to present problems and knowledge. In this paper we do not propose another interoperability framework but specify an architectural solution with an emphasis on technical and semantic perspective to solve interoperability problem between strategic and operational systems.

Architectural solutions for interoperability are aimed at dealing with system heterogeneity. Heterogeneity is a “double-edged sword”. On the one hand, heterogeneity allows the use of technologies that can solve the problem at hand with high efficiency. On the other hand, heterogeneity in data and knowledge systems is obstacle for the interoperation of systems [18]. To find a way out of this dilemma, architectural solution is required to balance heterogeneity and interoperability (e.g. [19]). Heterogeneity can be classified into paradigm heterogeneity (two systems express their knowledge using different modeling paradigms), language heterogeneity (two systems express their knowledge in different representation languages), ontology heterogeneity (two systems make different ontological assumptions about their
domain knowledge), and content heterogeneity (two systems express different knowledge) [18]. Bridging and homogenization (or linking and merging) are the alternatives mentioned in the literature to solve heterogeneity problems [7,20,21]. Homogenization is an a priori solution to deal with heterogeneity. It means that a given system resource (e.g. code of software), available in an initial format, will be translated into other format [21]. Bridging deals with heterogeneity by inserting a new system that will serve as an intermediate between inter-related systems. In our architecture a mix of strategies will be followed, a domain ontology will be developed for homogenization; and an inference engine and translation component will be used for bridging.

Typical examples of bridging solutions in IT are enterprise application integration (EAI) and service-oriented architecture (SOA). Those solutions mainly involve web services as this is the current widest accepted approach of integration, and business rules as this is the common approach in knowledge intensive organizations [22,23]. Orriëns et al. [24] have suggested that BRs can be used to create compositions out of available services. Yet, limited research of interoperability issues between BRs and web services can be found in literature. Some of them tried to interweave BRs into a BPEL frame [25], whereas others intended to build a framework without using BPEL, e.g. [26,27]. However, in those approaches, BRs and web services were not interoperable due to the use of different ontologies. D’Mello et al. [28] argued that business rule driven service composition requires BRs to be documented in a unified ontology.

The state-of-the-art research combines often both homogenization and bridging approaches to solve interoperability problem at technical and semantic level. This often results in the integration of semantic technologies into existing infrastructure, e.g. [12,29,30]. Combining semantic web service (SWS), business rule (BR) and software agent technology (SAT) is desired because each of them can contribute to a foundation for creating interoperability among strategy and operational processes. SWSs are an extension of web services with semantic descriptions in order to provide formal declarative definitions of their interfaces and functionality [31]. In this way, the web services can be automatically composed into an operational process. BRs can be used to represent strategy and policy in decision making as well as the formulation of operational processes. SAT can be used to construct a dynamic execution environment where operational processes are created and executed according to BRs.

A BR is a statement that defines or constrains certain aspects of the business. It is intended to assert business structure or to control or influence the behavior of the business [32]. BRs can be described in different formal or natural languages. The way to translate strategy and policy into operational processes often goes along with define, translate and deploy various BRs to convert knowledge into software system for execution. The execution of BRs has been studied within software agent environments (e.g. [33]). Software agents provide a way of structuring a complex process system around autonomous and communicative components. In strategy and policy implementation, many BRs originate from strategy and policy and influence process creation. For example, a strategy is to provide feedback to clients in their preferred ways, which depend on how they submitted their applications. This strategy results in a BR defining whether the process should use email or post for providing feedback. The consistency of those BRs has to be guaranteed to ensure the compliance of operational processes with strategy and policy. While some studies (e.g. [34]) proposed using SWSs and software agents for process composition, rare of them pay attention to the requirement of the compliance with strategy and policy. In our architecture, a technical solution to balance heterogeneity and interoperability between system components is provided for ensuring interoperability between strategic and operational processes. It combines SWs, BR and SAT to create business processes. The combination of these three technologies all together allows knowledge from strategy and policy to be used in process creation and execution directly.

Other related work can be found in software configuration management (SCM), which is the discipline of controlling the changes of complex software systems [35]. Typical SCM methods do not describe and explain the links between strategy and policy aspects that require certain functions delivered by a software system and the system itself [36]. Mohan et al. [37] proposed an approach to combine SCM with traceability to manage knowledge about the process of the development of software artifacts throughout the development lifecycle, which nevertheless does not explicitly address links to policy and legislation aspects. Apostoloiu et al. [6] presented a primary attempt to bridge the strategy and policy aspects in service systems. They proposed a software framework for representing changes in service models, finding inconsistencies caused by a requested change, generating and propagating changes to affected service artifacts, and alerting users about such changes. Yet they focused on the ontology models for decision making knowledge management. In our architecture, ontology model will be used for providing technical interoperability to allow process creation. It goes beyond the scope of SCM as strategy and operational processes are linked and three different kinds of technologies are integrated.

3. Research approach

Our research followed a design science research methodology by identifying problem, defining the objective of the solution with architectural principles, designing and developing the architecture components and relationship, demonstrating them by prototyping and evaluating the prototype. The steps are based on the six steps of the design science research methodology of Peffers et al. [38].

1. Problem identification and motivation: define the interoperability research problem and justify the value of an interoperable architecture as the solution. This was based on the practical problem of translating strategies into operational processes.
2. Define the objectives for a solution: the objectives of the architecture and accompanying architectural principles were determined in this step. Existing knowledge of what is possible and feasible based on the practical problem (as identified in the previous) was examined by conducting a literature survey. The architectural principles are used to constrain the solutions space in the following step.
3. Design and develop the architecture components and relationship: in this step the system components and their relationship are defined guided by the architectural principles.
4. Demonstrate the architecture: to demonstrate the architecture a prototype was developed and implemented. A scenario to demonstrate the architecture solution was developed.
5. Evaluate the prototype: experts were asked to observe and evaluate how well the prototype supports a solution to the interoperability problem.
6. Communication: this contains communication within the case study and in research outlets.

The architectural principles were derived from empirical studies of the problem. The architecture components and their relationship were developed according to those principles with further consideration regarding to the elements of a system, their functions and interoperability. Testing is an important part of any design science approach [39]. For the purpose of testing, a prototype is used to implement a scenario derived from a policy in the agricultural subsidizing processes for farmers (from Dutch
Ministry of Economic Affairs, Agriculture and Innovation). In the prototype, the related policy is translated into BRs and certain pre-designed SWs are composed into different processes according to the different service requests from clients. Software agents are used to model the disturbed IT infrastructure in the processes creation. Dynamic processes are automatically created at running time based on services requests and BRs. Qualitative evaluation was performed by asking experts about their judgment on the improvement in comparison to the existing situations based on time and cost criterion. Agility and interoperability are evaluated by the capability of dynamic process creation and fast process adaption based on quantitative measures of speed and cost. Although the architecture is demonstrated in a case study in which a policy is translated into operational processes the findings can be generalized to other areas having similar characteristics. This is what Yin calls ‘analytic generalization’ [40] (p. 38).

4. Architecture design

Our architecture aims to lay the foundations for an interoperable infrastructure for organizations in which vertical processes determine the implementation of horizontal processes. According to the ISO/IEC 42010 (IEEE 1471) [41] (p. 3) standard, an architecture is “the fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution”. TOGAF 9 [42] explains this concept and indicates the elements of a system architecture to be: (1) the principles and guidelines governing the design and evolution of the architecture over time, and (2) a blueprint of the system at the component level to guide its implementation. Based on this concept, our architecture should include the following elements that will be described in the next subsections:

- The architectural principles: the consideration of necessary design decisions to develop a concrete architecture; and
- The system components and their relationship: the components of the system and their functional relationship which describes how they interoperate to create business processes.

4.1. Architectural principles of an interoperable architecture

The use of architectural principles for designing service systems are commonly used in the design of systems [43,44]. Developing principles can be based on practice or by studying literature. The principles presented in this section derived from our empirical studies of the policy implementation in public organizations [3].

A literature review was conducted using the systematic scanning approach proposed by Levy and Ellis [45] and followed by information systems scholars (e.g. [46,47]). An extensive search of journals, conference proceedings and working papers was performed using ACM, IEEE Xplore, AISnet, ProQuest (ABI/Inform), Science Direct and Google Scholar. These databases contain the top information systems (IS) journals and conferences [45]. In the searching, we used the key words of ‘architecture principle’ or ‘architectural principle’ and ‘interoperability’. Although near 400 articles were found in the searching, most of them did not provide well-formulated principles but refer to the use of principles in guiding the design at high level. Those principles are mainly come from various reference frameworks (e.g, INTEROP NOE [48], Government of Canada Federated Architecture [49] and the NGOSST Technology-Neutral Architecture [50]) or well-known basic concepts (e.g. [51]). Based on this literature review an overview of brief principle concepts was made for further elaboration. It was our goal to cover a broad range of benefits and barriers rather than determining how often they were found. As a result we focus on three concepts for developing architectural principles with the consideration of interoperability: (1) modularity, (2) separation of concerns, and (3) representation.

As there are no sound architectural principles found but only generic concepts and concerns, a project team with 15 members was assembled to elaborate those concepts into architectural principles. The team members included 6 researchers from 2 two universities, and other members were architects either from public organizations or IT solution providers. The development process was a two round meeting each with steps of presentation, individual review and group discussion.

The Open Group Architecture Forum provide a template for describing principles, in which principles description by a brief explanation of the rationale and implications of following the principle [42], Aier et al. analyzed literature about principles and found that principles can include statement, rationale, implications, key actions, and measures and an extended definition taking the use and impact of an EA principle in its environment into account [52]. In Table 1 we summarize the three main principles of our architecture.

<table>
<thead>
<tr>
<th>Table 1 Overview of the principles.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Principles 1</strong></td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Statement</td>
</tr>
<tr>
<td>Rationale</td>
</tr>
<tr>
<td>Implication</td>
</tr>
<tr>
<td>Use and impact</td>
</tr>
</tbody>
</table>
service. A Business Service (BS) can be defined as a component that is assigned clear responsibilities and accountabilities, encapsulates a business function and can be reused in several processes. BSs can be viewed as a set of capabilities that can be reconfigured to meet changing objectives, e.g. [54]. They serve to map IT functionalities (e.g. functionalities implemented by software components and data structures) with activities (business structures necessary for process execution) [20]. BSs are loosely coupled and should enable the creation of dynamic business processes spanning information systems and even organizations.

Ideally, SOAs enable the on-demand composition of new business processes using existing services and minimizing the development efforts required to build new systems from scratch. As such, the creation of service compositions becomes an important issue. “Service composition combines modular services following a certain composition pattern to achieve a business goal, to solve a problem, or to provide new service functions in general” ([55], p. 30). An end-user service thus comprises a number of services, which in turn may be automatic services or composed services which might involve external services. Nevertheless, the execution might involve humans, for example when the invocation triggers a workflow to manually collect additional information and to make decisions based on criteria that are hard to automate. Human decision making can be modeled as a service executed by human. In this way a composite service is created and only exists while it is being executed.

4.1.2. Principle 2: separate business rules derived from policy from operational concerns

Organizations take the strategy and policy as input and implement it in their operational processes. Business rule approach is the common way for this purpose as it has stronger capability to deal with changes in the environment, in a comparison with graph-based approach (e.g. BPMN) [56]. The decision for solving paradigm heterogeneity has been made when the organization decides whether use rule-based or graph-based approach. For most knowledge intensive organizations, business rule approach is their choice [22,23]. Translating policies into business rules is done by deriving process design requirements from the policy descriptions [57]. However, BRs can be created from multiple sources. If once source is updated, only the BR related to these sources should be updated, and this should not influence BRs related to other sources. According to Goedertier and Vantienen [58], the following concerns can be included in defining BRs:

- The business strategy, internal directives, and procedures;
- Information prerequisites: the information required to start an activity; and
- Technical and common-sense constraints.

For example, in a subsidizing service for electronic branding (see the context in Section 5.1) the policy describes how to calculate a subsidy according to the number of sheep or goats. This calculation can be executed by using the BRs derived from the policy. Besides, it is a common-sense constraint that an applicant of this subsidy should be a farmer who already has a farm. Otherwise, the subsidy application might be a fraud. If the applicant is a new farm owner, a physical investigation of the farm might be required before the decision is made to provide the subsidy. BRs for checking farm ownership can be used to filter applications, but they are not derived from the policy.

Horizontal processes use business rules derived from multiple sources. Organizations have to identify the rules derived from policy in order to maintain legal compliance of horizontal processes. This can be done by deploying strategy and policy deriving BRs into decision services which are a type of business service. Taylor and Raden [59] used the term “decision service” to describe a self-contained callable component with a view of all conditions and actions that need to be considered in making an operational business decision. Decision services refer to policy, which implies that the use of these decision services is mandatory and they are the main services that can satisfy the service request in the final decision making. In the execution of the decision services, another kind of business service – supportive services – might be needed. In contrast, BRs in supportive services are not derived from policy but depend on other concerns of the organization, for example the BRs for checking farm ownership. Supportive services can be used as infrastructural services with high reusability. Services like information retrieval have a purpose to provide information/data for decision making. They allow a single information source to be reusable for multiple decision services. Other supportive services may facilitate decision services by providing user interfaces or report generation. The use of decision and supportive services allows the separation of BRs derived from policy and BRs derived from other concerns in horizontal processes. This has the advantage that if policies are changed only (the BR of) the decision services are affected and supportive services can remain the same.

4.1.3. Principle 3: distinguish between business rules at organizational, semantic and technical levels, and keep them consistent

Deploying BRs derived from strategy and policy into decision services separates them from other BRs in horizontal processes. Yet maintaining the compliance with strategy and policy requires the consistent use of BRs in vertical processes. Weigand et al. [60] distinguish the business and service execution environment, and suggested viewing the policy, formulation and execution of BRs as different levels. Interoperability is the clear connection between those levels. This refers to the bridging solution for solving language heterogeneity.

BRs have different forms depending on the level at which they are found: organizational, semantic or technical. Strategy and policy is focused on realizing desired behavior and fulfilling public values. Strategy and policy can be ambiguous, and different sources of strategy and policy might conflict. Therefore, they are not directly executable, need to be interpreted and often require extra information. The interpretation can be thought of as BRs at the organizational level [61]. BRs are contained in the interpretation in the format of a natural language.

For deploying the business logic and requirements in the ICT systems of the organization, the policy interpretation is further modeled through knowledge engineering [62]. At the semantic level, BRs are described using certain knowledge representation technology. The recent development of BRs in the Semantic Web domain focuses on a standard representation of BRs, e.g. the rule interchange format (RIF) from W3C. Although these kinds of BRs are computable, they are not directly operated in a business process. Namely, they are not directly used in process creation and decision making during process execution. The BRs directly used in process creation and decision making during process execution are maintained at the technical level. There are two kinds of BRs used during process execution. The first involves operational rules (e.g. condition-action (CA) rules) which are involved in the execution of decision services. The other involves declarative rules which describe the process logic. Connecting the above mentioned BRs and maintaining their consistency is the key to creating interoperability between vertical and horizontal processes.

In practice there is a need to cope with different sources of strategy and policy. The use of decision services allows BRs derived from different sources of strategy and policy to be encapsulated into
different decision services. Using different levels (organizational, semantic and technical) results in clearer separation, but creates language heterogeneity which results in the risks of having inconsistent rules, as BRs at different levels have different forms. The challenge is to connect knowledge representation models to BRs used by horizontal processes. The rule interchanges between levels refers to our previous research [63].

4.2. Architecture components and relationship

The architecture components and relationship, the blueprint of the system, are designed based on the architectural principles. The components comprising the architecture have different functions, supporting interoperability at the semantic and technical levels. We consider the organizational level, in which law originates from multiple sources and in which processes are executed by independently operating departments, to be given and dependent on the context (see, for example, the case study background in Section 5.1). Next we will discuss the semantic and technical level.

4.2.1. Semantic level

At the semantic level, there are three knowledge repositories: the domain ontology for describing unified vocabulary and taxonomy, the SWS description for semantically presenting web services, and the BR model for semantic representation of BRs.

Domain ontology has been indicated as necessary for interoperability. A domain ontology refers to the knowledge elements that comprise the conceptualization of the domain in which the architecture is going to be applied [30]. A domain ontology shared by all the components can avoid conflicts and mismatching on concepts, because in a well-defined and unified ontology concepts are unified without overlapping each other. Although it is possible to present BRs in an ontology, a domain ontology is essentially different from BRs. As its definition states, the purpose of a BR is to influence or guide process behaviors. In contrast, a domain ontology provides vocabulary and taxonomy (or a concept hierarchy) to facilitate communication. It does not relate to the content of the message, and therefore, it does not relate to BRs. But the vocabulary provided by the domain ontology can be and should be used by the description of BRs.

SWS description semantically describes web services. Dietze et al. [64] claimed that "a SWS description is formally represented within a particular ontology that complies with a certain SWS reference model such as OWL-S or web service modeling ontology (WSMO)” (p. 248). Such a description is suitable for both service supply and request. All relevant aspects of a web service should be described, including the distinguishing of a decision service or a supportive service.

BR models are the semantic representations of BRs. In our reference architecture we focus on two kinds of BRs. The first kind involves operational rules which are translated into executable rules used in decision making and involved in the execution of decision services. The other kind of BR concerns declarative rules which describe the process logic. The most important role of these BRs is to clarify the relationship between service requestors and decision services. Both these two kinds of BRs are derived from policy and should maintain traceability and consistency with their policies.

The above-mentioned knowledge entities should be presented in proper formats that allow an inference engine to perform reasoning operations on them. The engine allows the agents to use the knowledge presented through semantic technologies and to use different forms of BRs to achieve the connection. The selection of an inference engine thus depends on the technology used in knowledge representation. For example, Pellet [65] is used to infer the domain ontology which is described in OWL 2. If necessary, a rule format translation component should be involved, for example a component to translate RIF rules into BDI agent rules [63]. For the BRs that are encapsulated in web services, a translation component is also useful for the construction of the web services, for example by generating codes to assist in their development.

4.2.2. Technical level

According to their functionality in process execution, there are three types of agents for creating processes: operational, process rule manager and resource manager agents. An operational agent is an execution entity that performs tasks in process execution. By representing its role as a service request, it invokes decision or supportive services to finish the tasks. The operational agents, decision and supportive services are the building blocks directly involved in a running business process. For a clear process control, we locate the decision and supportive services according to the typical horizontal administrative process pattern [66], which includes including the following steps: (1) submission and registration, (2) information gathering, (3) filtering, (4) decision making, and (5) notification and issuing. By using this pattern, the operational agents already know with which agent to communicate for the next step in the process. They only need information of what services they should invoke.

Process rule and resource manager agents act as coordinators by using the knowledge from three knowledge repositories via the inference engine or translation complement. A process rule manager agent provides knowledge about the decision services that an operational agent can invoke. The resource manager agent provides information about available supportive services to facilitate the process creation. The process creation is performed during its execution by operational agents invoking necessary services. Services are shared among operational agents which carry similar or different end-user requests. An overview of the architecture can be seen in the following figure (Fig. 2).

5. Prototyping and testing: agricultural subsidizing processes

The goal of the testing was to find out whether the architecture is interoperable by analyzing whether new policies stored in policy/strategic systems (vertical processes) can be automatically translated into the workflow systems (horizontal processes). The domain was selected based on the need to adapt frequently and for providing access to the information needed. The test of architectural principles was done indirectly, by developing and evaluating a prototype within an organizational context; and directly, by asking experts to provide their opinions about the value of the principles. We performed the following steps:

1. Build domain ontology
2. Define business services
3. Define business rules
4. Test dynamic process creation

The Dutch Ministry of Economic Affairs, Agriculture and Innovation (Ministerie van Economische Zaken, Landbouw en Innovatie, or EL&I) is responsible for promoting sustainable economic growth in the Netherlands and collaborates with farmers toward more sustainable agriculture. The Office of Arrangements1 takes part in the policy-making surrounding agriculture and nature. It provides a variety of services in the fields of: manure management; financial, subsidy and EU regulations; permits and

---

waivers; the identification and registration of animals, parcels and relations; and crisis management. All of these services need to be frequently changed due to changes in agricultural policies. Implementation is often delayed due to a lack of interoperability between policy and workflow systems used for service provisioning. The systems use for supporting the vertical strategy formulation process and the systems handling the execution of operational processes are not able to communicate with each other. Recently there was a change in their strategy which aimed to enhance the speed of its policy implementation and the policies should help to reduce the administrative burdens for the sector and the government.

Working as a window of EL&I is an important feature of the Office of Arrangements, as it is also a central Customer Contact Center, where farmers, intermediaries and others employed in the agricultural sector can go to if they have any questions and/or remarks. The Office of Arrangements is an agency of the Netherlands as well as an agency of the EU and has branches in Assen, Den Haag, Deventer and Roermond. With the help from experts at the Office of Arrangements, a policy was selected for prototyping, and business services and business rules were determined by interviewing domain experts. The experts also verified the results to ensure their correctness.

5.1. Background

To demonstrate the architecture, we selected certain technologies to implement each components of the system. Yet, the architecture does not stick to particular technologies. For developing the domain ontology model, we use OWL as the modeling language. OWL, from W3C, is currently the most popular ontology representation language [67]. OWL 2 describes the domain in terms of individuals, classes, properties, data types and values. It consists of a set of axioms and facts that describe the domain. Ontologies that are described in OWL 2 can (and generally are) stored as Web documents and can be combined into larger collections of information. Protege 2 is used as the development tool for editing OWL and OWL 2 ontologies.

As modeling languages for SWS description, OWL-S [68], formerly DAML-S, was selected. This is an ontology for the description of semantic web services expressed in OWL, and a core set of markup language constructs for describing the properties and capabilities of web services in unambiguous, computer-interpretable form. Its inherent compatibility with OWL makes it a reasonable choice. Tool development for OWL-S has largely taken place as separate elements or components developed by different research groups on a stand-alone basis rather than as a full-fledged framework covering the entire life cycle of Semantic Web applications [69]. Although there is an OWL-S Protégé-based Editor plus-in available, it only works for Protégé 3.2 or earlier versions, in which OWL 2 was not supported. This means the developer either uses OWL 1 and earlier versions of Protégé together with OWL-S editor plus-in, or uses OWL 2 with current version of Protégé and other stand-alone editors like the OWL-S Editor 3 and ASSAM [70].

The current development of business rule standards allows them to be described using semantic technology. The rule interchange format (RIF) became a W3C recommendation in 2010. For declarative BRs which clarify the relationship between service requesters and decision services, we used RIF as the technique to represent BRs because of its compatibility with OWL.

Lying between the knowledge representations and the multi-agent system, the engine must be able to do the reasoning on the knowledge repositories. The currently available inference engine that satisfies most of the above requirements is Pellet [65]. Pellet is an OWL 2 reasoner. It provides standard and cutting-edge reasoning services for OWL ontologies. Pellet is viewed as the leading choice for systems for which sound-and-complete OWL DL reasoning is essential [71]. It includes support for OWL 2 profiles including OWL 2 EL. However, in terms of rule supporting, Pellet does not support RIF. Instead, it supports reasoning with SWRL [72] rules. SWRL is a rule language which was designed as an extension to OWL. It is one of the simpler rule languages, and most SWRL features are covered by RIF-BLD, with the exception of

2 http://protege.stanford.edu/.

---

Fig. 2. Interoperability architecture for policy implementation.
“different-from”, thus it should be possible to exchange most SWRL rules via RIF [73]. In fact, we were not able to find any reasoner that currently supports RIF. RIF is quite a new standard; the industry and academia still need time to develop supporting tools for it. It is not clear whether Pellet will include support for RIF in the future. But the compatibility between RIF and SWRL allowed us to choose Pellet. In the development we used Protégé 4.1, together with Pellet Reasoner Plus-in version 2.2, to build the prototype.

5.1.1. The context

One highly volatile topic in the agricultural field is the subsidizing of the electronic branding of sheep and goats. The objective of this policy is to encourage farms to use electronic branding under the financial capability of the government. The subsidy policy frequently changes due to the government financial situation and changes in other agricultural policies and strategy. In this study, we selected a 2010 version of this policy. Reopened in 2010 from the original ministerial regulation “CAP Income 2006” (GLB-inkomenssteun 2006\(^4\)), the following policy is in part about the opportunity to receive support for the electronic branding of sheep and goats. For the purposes of this paper we only list a translation of (a part of) Article 38, 38a and 38b. This policy has 78 articles in total and includes a much broader scope with more related concepts and legislation.

**Article 38**

(\(\ldots\))

b. Sheep or goat: sheep or goat, as defined in Article 1, subdivision q respectively subdivision r of the Regulation identification and registration of animals;

c. Electronic brands: identifying a sheep or goat, according to Article 12e, first paragraph, Sections A through N, of the regulation 'Identification and Registration of Animals'.

**Article 38a**

1. The Minister shall provide one-time special assistance to farmers in the form of a financial contribution for the electronic branding of sheep and goats. 2. Specific support amounts to \(\varepsilon\)1 per sheep or goat, taking into account that this amount is reduced proportionally for all reimbursement requests to be considered if the total of the requests that should be taken into account for compensation exceeds the amount of \(\varepsilon\)2,000,000.

**Article 38b**

1. The assistance, referred to in Article 38a, first paragraph, can only be requested by farmers who were owner or tenant on May 15, 2010, of an agricultural company that is assigned to a UBN and on which, on the basis of Article 37, first paragraph, of the regulation ‘Identification and Registration of Animals’, as it read on November 1, 2009, it was indicated that based on UBN at that moment more than 100 sheep or goats were kept. 2. The Minister shall only provide support for the electronic branding of sheep and goats that were born on or before December 31, 2009. 3. The Minister shall provide support only if 90% of the animals born before January 1, 2010, which are present at the UBN on June 30, 2010, are electronically branded or before June 30, 2010. (\(\ldots\))

Simply speaking, the above policy describes a number of relevant legal rules, which can be reformulated in the following.

- Specific support may be provided only to farmers who keep more than 100 sheep or goats;
- The amount of specific support is re-calculated and reduced proportionally if the total request exceeds the amount of \(\varepsilon\)2,000,000;
- Specific support is only provided for the electronic marking of sheep and goats that are born on or before December 31, 2009;
- Specific support may be provided only if 90% of the animals are electronically branded by June 30, 2010;
- Specific support should be provided if the above conditions are met; and
- Only the Minister can provide specific support.

In this illustration we intentionally hide the complicated preconditions and calculation methods of the specific support to avoid the explosion of involved legal concepts. In the rest of this chapter we will describe how such policy can be implemented with our architecture. The pilot implementation is a system that regards farmers as end-user and provides them a convenient way to gain subsidy for their use of electronic branding. Currently the organization provides the subsidizing service in a semi-automatic way, where some segment sub-processes and components isolated. They have their CRM application to manage clients’ profiles, risk control application to identify clients with rotten records, and an expert system to help in the calculation of subsidy. Yet those components are not well interacted and many

connections are manual and paper-based tasks. In our prototyping, a pilot system integrates those segment sub-processes and applications into a service delivery process for clients. We use the following process diagram written in the Archimate modeling language to demonstrate the steps of a subsiding process and its interactions with clients (Fig. 3).

5.2. Building a domain ontology

The above policy refers to many legal concepts, such as “sheep”, “goat”, “animal” and “specific support”. Fig. 4 is an illustrative domain ontology to describe the concepts used in this case study. Due to space reason, many issues fall outside the scope of this ontology: for example, the different kinds of farmers (if it is necessary to distinguish them), the departments of EL&I, the different subsidies (if they are all relevant to electronic branding of sheep and goats), and the different kinds of sheep and goats (if this impacts the gain of specific support). Moreover, the data property of an entity (e.g. “person” has “ID” and “account”; “farmer” has “sheep number”, “goat number”, “sheep electronic brand number” and “goat electronic brand number”; “sheep” and “goat” have “date of birth”; “electronic brand” has “date of branding”, and “result” has “amount” as the result of calculation) might eventually need to be included into the domain ontology for operational reasons.

5.3. Defining business services

The process that allows clients to apply for subsidy for the electronic branding of their sheep and goats is composed by several business services. BSs are divided into decision and supportive services, as shown in Table 2. Decision services hold a tight connection with the policy, and the connection between the policy and the business rules in execution should be traceable. Decision services ensure that the processes comply with the policy. Supportive services allow for the reusability of resource among different processes.

Both decision and supportive services are described with semantics. The service description is formally represented in a format specified as [service functionality, input set, output set] which complies with a certain SWS reference model such as OWL-S. Eventually, the services will be described in OWL-S format with the help of proper development tools.

BSs are derived using the criteria of Parnas [53], who suggested that each module should hide some design decision from the rest of the system. We assume that the system needs a decision service (DS01) to calculate how much subsidy is available for a client. We also need another decision service (DS02) to allow manual investigation of special cases. To start the process, the system needs at least one supportive service (SS01) for submission intake. Alternative intake services can be issued to allow multiple submission channels (e.g. one for electronic channel, and one for front offices). We also need a supportive service (SS02) to check via the UBN system the number of sheep and goats born before January 1, 2010 that belonged to the client and how many of them were electronically branded by June 30, 2010. Similar information gathering services can be issues for e.g. checking client information from a CRM system. At least a filtering service (SS03) is needed to categorize submissions for automatic decision making or manual decision making. This step can vary depending on the organizational strategy, e.g. using a simple black list or some complex risk models. The EL&I currently uses a complex risk control model concerning the historical behaviors of the client. A high risk will trigger manual investigation for decision making. Furthermore, we need one supportive service (SS05) for bank transactions.

5.4. Defining business rules

BSs that are used in the decision service are separated from the BRs that describe the process logic. Rules used in decision service are closely related to the policy. Being traceable to the piece of legal source text allows easy maintenance of them. Many expert systems can fully fulfill such a requirement. In our prototype the rule models are represented in RIF-Core. An example of rules in decision service is given as follows. This rule strictly responds on the calculation method described in the policy. Note that most of the IRI s used in the examples of this paper are fictitious and do not represent real entities. To improve the visual appeal, the example takes advantage of the syntactic shortcuts (allowed by the RIF Datatypes and Built-ins specification). For instance, $100$ is used instead of $\text{"100"^^"xs:integer"}$. For the same reason, we directly use the mathematical and binary relations operations. For example, in practice, $7a > 10$ should be defined as External(pred:numeric-greater-than(?a 10)), where pred is the

![Fig. 4. Illustrative domain ontology in the CAP case (created using Protégé).](image-url)
prefix defined as Prefix (pred <http://example.com/rif-built-in-predicate#>).

Prefix(mes <http://example.com/message#>)
Forall ?result {
  If And(?result # mes:Result
    ?sheep_number + ?goat_number >= 100
    (?sheep_Electronic_Brand_Number + ?goat_Electronic_Brand_Number) >= 0.9
  )
  Then Do( Modify(?result[result:amount-> 4 * (?sheep_number + ?goat_number)])
}

5.5. Policy implementation

In the final step, the implementation of changed policy is demonstrated by dynamically creating processes based on the changes made to it. We tested the creation by implementing two types of changes in policy occurring several times the last couple of years:

- The changes in calculation method: the criteria of subsidy calculation are changed from year to year; and
- Exceptions that require manual decision making: not all the submissions are standardized; some special submissions require human intervention and this happens frequently.

Below is an example of such a process creation.

Fig. 5 shows that the end-to-end process is supported by the architecture. Departments can continue operating autonomously, have resource proprietorship, and operate in their existing hierarchy.

5.6. Evaluation

The results were discussed with the Office of Arrangements. In its opinion, the architecture creates the interoperability between the previous disparate policy and operational workflow systems and is able to adapt to changes in policy by using the output of the vertical system as input for adapting horizontal processes. The following main advantages were viewed:

1. Agility: some new policies (e.g. updating the criteria of subsidy calculation) can be implemented immediately in decision services, whereas others need extensions/modifications of the supportive services (e.g. adding new subsidy policies for the electronic branding of animals other than sheep and goats). Changes in the calculation criteria are capsulated in the decision service DS01. This even allows for parallel decision services, as different versions of a single policy that are validated in different years. “At the first glance, it already indicates where the horizontal process will be changed,” one of the interviewees said. Although adaption within a certain service can vary, finding out the impact from changed policy on horizontal processes is easier than before. This reduces the uncertainty in policy implementation.

2. System flexibility: new extensions, such as the risk management task executed by a subsystem, can be easily integrated into the horizontal process. An interviewee commented that “risk management is another kind of important information and knowledge in the routine processes but is not mentioned in the policy”. The EL&I has made use of its cumulative knowledge of risk management for many years. This is the important internal strategy that strongly impacts the horizontal process design. The architecture is able to support this consideration.
developed knowledge commented that the output of the vertical processes is directly translated into horizontal processes, compliance to policy is ensured.

4. Implementation cost: as the architecture allows quick implementation and easy adaption for changes, less FTEs is required. This results in cost reduction of new changes, after the upfront costs of having the architecture in place have been made. Current operational process in the organization connects each step manually, as each of them is performed by a siloed department. Although ICT applications provide support for the business processes within each department, interoperability between them is hardly achieved. Manual documenting and communicating on cases is a daily job. The architecture provides a solution for achieving interoperability resulting in a reduction of a number of manual tasks.

As the architecture is based on three main principles, the value of the principles is also reflected in the testing. Principle 1 suggests the use of business services. This eventually results in the agility to respond changes in policy. Principle 2 suggests the separation of business rules from different sources. This facilitates the maintenance of the compliance to policy and also enables new extension. Principle 3 suggests the consistency of business rules at different levels. This eventually ensures the compliance of operational processes to the policy. The interviewees all agreed on the benefit of those principles.

There were also a number of disadvantages mentioned, related to complexity, knowledge and process. Knowledge about ontologies is required; especially policy-makers find it difficult to build and maintain a complex ontology. Furthermore, a closer cooperation between policy-makers and business process implementers is required. Reusability of decision services is limited, as they are developed for supporting specific processes. Some interviewees commented that some filtering of submissions also requires knowledge from decision services.

6. Conclusion

Whereas most research in interoperability is focused on horizontal (workflow) processes, little research has been done on vertical interoperability to enable communication between disparate strategic and operational systems. The outcome of vertical direction-setting processes influences the horizontal or operational processes. A lack of interoperability between those two processes results in long delays and high costs in the implementation of new strategies and policies. Traditionally strategy and operational processes are separated and viewed as two different of a kind, whereas today’s fast changing environment requires interoperability to ensure that changes can be quickly implemented. We have presented an architecture which lays the foundations for interoperability between vertical and horizontal systems and processes. This architecture is based on three main principles: to define business processes using Business Services; to separate business rules derived from policy from other concerns; and to distinguish between business rules at the organizational, semantic and technical levels while at the same time keeping them consistent. Whereas oftentimes principles are not tested, in our research the architecture is tested in a prototype using scenarios in which policies are directly implemented in horizontal business processes. The testing shows that the architecture helps to implement some policies immediately and others in a faster manner than previously possible, in this way contributing to the agility of policy implementation. The architecture has the additional advantages that the business processes comply with strategy and policy, new elements can be included more easily, and the cost can be reduced. As the principles are the foundation of the architecture, the testing also reflects that those principles are useful for improving interoperability.

In the system architecture the three knowledge repositories—domain ontology, semantic service description and business rule models—are used for creating semantic interoperability among SWSS, BRs and SAT. By combining these technologies interoperability between vertical and horizontal processes is created. SWSS are used for semantically describing services, software agents execute the service composition by invoking SWSS, and BRs translate strategy/policy statement into executable rules for decision making and operational processes creation. Although the architecture achieves fast policy implementation at low costs and ensure that operational processes comply with the strategy and policies, there are two drawbacks: defining and maintaining the domain ontology is resource-intensive, and the use of the process pattern limits the reusability of decision services.

The architecture captures the technical and semantic level and the organizational level was left out. Connecting vertical and
horizontal processes requires governance mechanisms to make it work. At the organizational level there also exist factors influencing agility and interoperability which relationship was not investigated. Related to this, a limitation is that we only evaluated the architectures within a certain setting using a single type of policy. Within these setting analytical generalizations is possible, however, it might be that within other settings or with other strategic directions the architecture might not provide the necessary interoperability. In particular when large changes requiring a drastic change of operational processes, the interoperability provided by the architecture might be sufficient. The boundaries of the usefulness of the architecture should be further investigated.

References


Dr. Yuwei Gong is a researcher at the Information and Communication Technology section of the Faculty of Technology, Policy, and Management at Delft University of Technology, the Netherlands. His research interests include IT Architecture, Business Process Management and Service Engineering. He received a Ph.D. in Information Systems (2012) and a M.Sc. in Computer Science (2008) from Delft University of Technology. During his research, he worked directly on the AGILE project funded by the Netherlands Organisation for Scientific Research (NWO) and provided consultation to the Immigration and Naturalization Service (IND) for business process improvement. He has been involved in several international conferences as a program committee member or reviewer, including IADIS Information Systems Conference. Contact him at y.gong@tudelft.nl.

Prof. Dr. Marijn Janssen is full professor in ICT and governance of the Technology, Policy and Management Faculty of Delft University of Technology. He is Director of the interdisciplinary Compliance & Design Management Master programme. His research interests are in the field of infrastructures, public-private networks and his research focuses on orchestration, shared services, intermediaries, open data and open government. He has been a consultant for the Ministry of Justice and received a Ph.D. in information systems (2001). He is associate editor of Government Information Quarterly (GIQ) and the International Journal of E-Government Research (IJEGR), serves on several editorial boards and is involved in the organization of a number of conferences, including IFIP EGOV. He has published over 260 refereed publications. For more information visit www.tbm.tudelft.nl/marijnj.