A Systemic, Ontology-driven Approach to e-Services Governance

Bill Karakostas[1] and Yannis Zorgios[2]

[1]Centre for HCI Design, School of Informatics, City University
London EC1V 0HB, U.K.

[2] CLMS (UK) Ltd, 73 Park Lane, Croydon, CR0 1JG, U.K.

Abstract. This paper proposes an ontology driven, systemic approach to e-services governance. E-service governance refers to frameworks and policies for controlling the development and provision of e-services within an organisation. Given the level of complexity of current e-services, it is necessary to think of them as systems of interconnected elements that are more complex than the sum of their parts. Our IDEF0 based, system theory inspired, modelling approach, captures the essence of governing systems of e-services contained (recursively) within higher order systems. We use ontologies to represent explicitly systemic properties of services such as context, control/constraints and feedback. Governance rules constrain the syntactical, semantic, and behavioural properties of service ontologies and, due to the hierarchical ordering of service systems, can be applied to e-service portfolio management, architectural design compliance, and runtime SLA enforcement. Ontology mapping capabilities allow governance rules to be described using concepts appropriate for the different levels of service.

1 Introduction

SOA governance is an extension of IT governance that focuses on the life cycle of services and composite applications in an organization's Service Oriented Architecture (SOA) [6] The purpose of SOA governance is to establish and maintain the essential relations between the controlled services system and the larger (controlling) organisational system [4].

According to [12] governance policies can affect every aspect of the service lifecycle, including design, deployment, and operation. Governance can therefore be classified into the following categories:

– *Service Identification Governance***,** driven by organisational considerations, market analysis, stakeholders, policies, objectives etc.
– *Design Governance***,** i.e. rules and constraints to drive top down decomposition of services
– *Runtime Governance* that controls service execution via business environment oriented feedback
– *Asset Governance***,** i.e. organisation policies for reuse of services assets

– *Management governance* that *r*efers to policies for adaptation, modification, retirement, etc. of services

According to [4] the main benefits of e-services governance are:
- Alignment of e-services to business needs.
- Adaptability and integrity of e-services.
- Ability of IT services and organisation systems to co-evolve effectively.
- Enhancement of the business value of services.

From a service lifecycle perspective, governance rules fall into one of the following three categories:

Portfolio Governance Rules: These refer to business modelling for SOA. They are used to maintain associations between services and other asset types of the organisation.
Architectural Governance Rules: These rules ensure that services have been defined according to the enterprise's SOA infrastructure. For example: does a service has a higher level (business service) that defines its scope?
Behavioural Governance Rules: These rules constrain the runtime behaviour of a service. These rules are used to enforce Quality of Service (QoS) and Service Level (SLA) agreements at runtime.

This paper presents an approach to e-service governance that is influenced by Systems theory as well as more recent work in using ontologies, to describe properties of e-services. The paper is organised as follows. The next section discusses e-service governance from a system theoretic perspective. Section 3 introduces the IDEF0 notation that allows the description of hierarchical systems of e-services. Section 4 formalises the concept of governance using hierarchical systems of ontological descriptions. In Section 5 the concept of ontology based governance is illustrated for different phases of service engineering, and with different types of ontologies. Section 6 outlines an approach to runtime governance using the concept of feedback. Finally, section 7 discusses related work and suggests future research activities.

2 Systemic Principles of e-Service Governance

Systems theory, pioneered by von Bertalanffy, has been widely used as a framework for understanding the nature of computer systems. In its origins, systems theory was motivated by the desire to find a way to consider entities not as isolated beings but related to their environment, yet maintaining a state that is not wholly at the mercy of the environment, and so maintaining integrity as a distinct organism.

Systems theory views a system as having a boundary separating it from the environment and across which interactions occur, and within the boundary are the subsystems of which it is composed. Central to systems theory, is the concept of hierarchical combination of sub-systems into systems, and their interaction with a wider system that is the environment. Under this view, the environment may then be seen as a wider system of which the system is a subsystem. By recursive application, this rule yields a part-whole hierarchy. In the computer science domain, systems theory depicts a computer system as composed of subsystems, sub-subsystems, and so on. This provides considerable elegance in analysis, since the analyst may apply the very same method of thinking to any layer in the hierarchy. However, an important aspect in systems theory is that the hierarchy is composed of layers that demand different levels of description [2]

Under the systemic perspective, services are hierarchical systems comprising organisational (i.e. people, non IT systems, physical resources) and IT systems, that automate in part or totally (i.e. as in the case of Web services) the service delivery. A business service is a system of systems, as it comprises multiple heterogeneous, distributed business and IT systems that are interlinked as networks at multiple levels and in multiple domains. As services are provided by the manipulation of organisation resources by processes, they are themselves systems embedded within larger systems and ultimately within the organisation system. Thus, the principle of governance applies in a recursive manner i.e. across the organisation, its services, and their constituting parts.

Under this premise, service governance must be applied across several hierarchical layers of business structures, IT architectures (e.g. SOA), IT systems, and environments, down at the level of executable software services, for example Web services (Table 1). Because, however, as argued above, different levels in hierarchical systems typically require different levels of description, service governance may need to be applied differently at each level. But, since service governance is essentially about top down control, a fragmented and disjointed approach to governance will fail to bring benefits such as alignment of e-services to business needs and co-evolution of business and IT services.

Thus, the paper proposes a holistic approach to service governance that is driven by a hierarchical systemic view of services.

Table 1. e-Service hierarchy levels.

The organisation context
The business system level
The SOA level
The Web service level
The technological infrastructure level

The conceptual tool to implement this approach is provided by the IDEF0 systems modelling methodology [5]. In particular, we exploit two principles of IDEF0:

– the *hierarchical decomposition* of systems, and
– the concept of *control* which can capture the semantics of governance rules

We employ the above modelling principles to apply governance rules at different levels/systems, and to enforce their consistent application across the service systems hierarchy. However, as argued by [12], to achieve consistency and automated conformance checking, dynamic discovery binding and enforcement, such governance rules should be in a machine-usable format .We additionally propose, therefore, that ontologies can be used to describe governance rules in a machine interpretable format. As different levels of the service hierarchy require different representations, we employ multiple ontologies and ontology mapping mechanisms. Indeed, this is the approach explored in the following sections of this paper.

3 The IDEF0 System Modelling Methodology

The IDEF0 system modelling notation [5] has been employed in systems analysis to model systems of any type, as hierarchical ordered networks of atomic elements *called processes* (and also *functions* or *activities*). IDEF0 introduces the concept of 'control' in a system model. The control depicted by the top entering arrow in a 'process box' is used to describe how the performance of the process is constrained by specific rules, guidelines, or conditions that are required to produce the correct output (Figure 1).

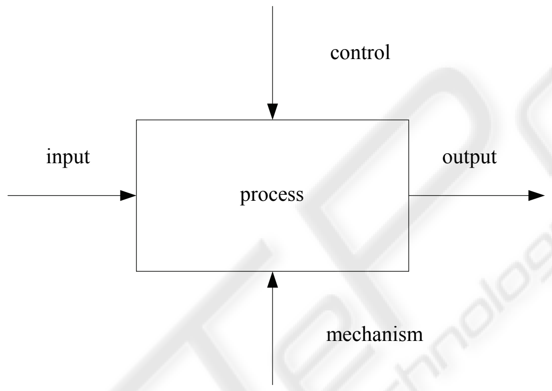

Fig. 1. IDEF0 notation.

The principle of hierarchical decomposition in IDEF0 suggests that a process can be decomposed into a network of processes (typically 5-7). This decomposition can be carried out over several levels (Figure 2).

Processes, at any level of decomposition, may have controls that are inherited from the parent process, in addition to any new ones introduced at that level. IDEF, though, has not formalised the concept of control, defining controls simply as conditions required for the function to produce correct outputs. Intuitively, the concept of control inheritance means that the conditions required for the correct operation of a function must also apply to the correct operation of its children functions.

In the context of service governance, the concept of control is suitable to be used for representing governance rules. Thus, informally, each service concept must be controlled by at least one governance rule. That rule specifies conformance conditions for the service. For such conditions to be meaningful, however, they must refer to system variables that can be monitored and measured, at that particular level of decomposition, i.e. to service constructs, such as inputs, outputs, mechanisms (i.e. resources) and their inter-relationships.

Of course, when a control (governance) rule is inherited from a higher level, it is possible that the rule is originally expressed using concepts applicable to that (higher) level. Therefore, because the rule might refer to the state of affairs of a different universe of discourse, it may not be possible to test its validity in the current context. We may need therefore to resort to model mapping/transformation techniques in order to translate governance rules for each level of service system decomposition.

These ideas are formalised in the following section.

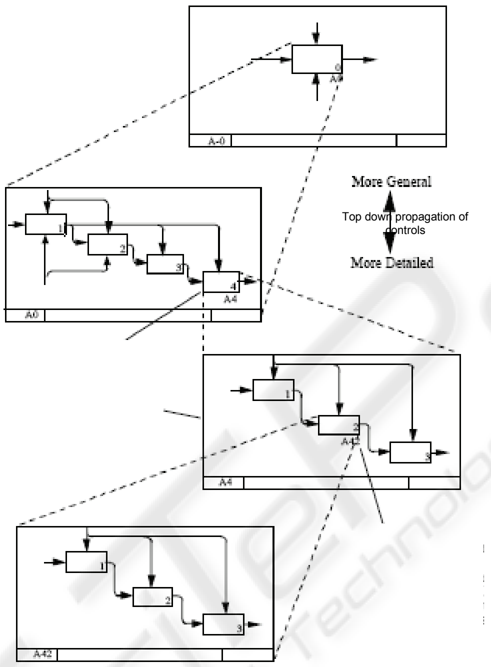

Fig. 2. Hierarchical decomposition of services.

4 Formalising Governance using Ontologies

As argued above, to achieve automated governance conformance checking, both governance rules and service models must be expressed in a machine interpretable notation. Ontologies provide us with this ability, i.e. with machine interpretable models of a of discourse (domain). Here we define a service governance ontology *O* as a tuple *O* := *(C,R, H,G)* where

– *C* are service concepts,
– *R* are intra-level concept relationships
– *H* are hierarchy decomposition (inter-level) relationships that specifying how service elements are decomposed (refined) over a succession of levels, and
– *G* is the set of governance rules.

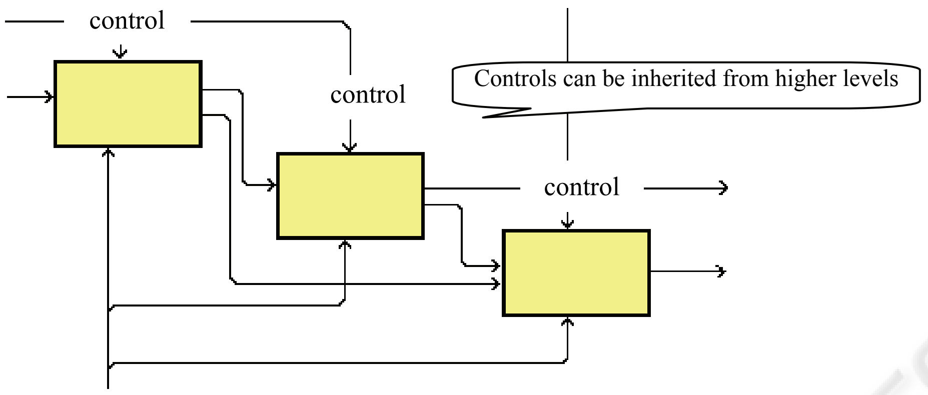

Fig. 3. Inheritance of controls.

A governance rule $g_n \subset G$ (defined at level n of the hierarchy) is a logical relation of a set of concepts $c_n \cup R_n$ where $c_n \subset C$ and $R_n \subset R$.

Assume that at decomposition level m (where m > n) the set of concepts c_m is used to describe the state of affairs, and that m inherits g_n from level n. To be able to test if rule g_n is satisfied at level m, we need to infer all rules G'_m from $g_n \cup H_{nm}$ where H_{nm} is a subset of H, i.e. a set of relationships relating concepts belonging to $c_n, c_{n+1}, \ldots, c_m$, and then check all rules $g_m \in G'_m$.

As an example, consider a governance rule, defined at level n, stating that for all process p, the mechanism f used to implement p must have been certified for quality assurance. Assuming also a 'realises' type of relation, i.e. a partial mapping from m to n:

$\forall f, p$: Implements(f, p)$\Rightarrow$ certified(f).
$\exists p'$: realises (p', p)
$\exists f'$: implements(f', p')

infer rule

certified(f')

The above inference states that the mechanism f' used to implement process p' must be certified. Of course, the concept of certification may have different meanings, depending on the level of the service hierarchy (i.e. business process certification vs. software process certification). This governance rule can be enforced at the (SOA) design phase. Other governance rules applying to the runtime phase are discussed in subsequent sections.

5 Applying Ontologies to the Governance of e-Services

Ontologies, therefore, can be used to represent formalised governance knowledge, and to actively assist in consistency checking, monitoring, and enforcement. Because of our hierarchical systemic view of services, ontologies can be used to express governance at the following phases:

– **Service Identification and Definition Phase.** Ontologies at this phase represent organisational goals and policies, resources and other critical assets. Governance rules express fundamental axioms relating services to higher level business concepts, such as, for example, that each service must be associated with at least one organisational goal or objective.
– **Service Design phase,** where method rules are applied to drive top-down decomposition of services. Resource ontologies can be used to state such rules. These can be embedded in a service design tool, integrated with an ontology editor, to automatically check decomposition/inheritance rules in service design.
– **Service Lifecycle Phase.** Ontologies here are used to express rules that govern and enforce lifecycle policies.
– **Service Asset Management Phase.** Ontologies here can be used to codify policies for management (cataloguing, reusing, retiring) elementary/atomic services.
– **Service Runtime.** Ontologies at this phase are used for monitoring and service execution control. These can be embedded in the service runtime execution environments. Constraint satisfaction systems, or theorem provers, can be used to enforce for example SLA constraints on executable services At runtime, all governance rules that are directly applied to the executing service or inherited from ancestor levels are evaluated.

Ontology Mapping and Interoperability

When moving across hierarchy levels, we might need to translate between ontologies syntactically or semantically. Ontologies might also need to be interoperable, for example, a personal ontology must interoperate with a domain ontology, as in Table 2.

In general, mapping between ontologies at different levels of the IDEF hierarchy can be achieved in several possible ways i.e.

- Via inheritance of concepts
- Via integration
- Via concept mapping. Ontology Mapping is the process whereby two ontologies are semantically related at conceptual level, and the source ontology instances are transformed into the target ontology entities according to those semantic relations.

In this approach we employ the concept mapping approach, where concepts from different levels of the hierarchy are related with hierarchical 'implements' type of relationships. This is not an 'is-a' (taxonomy) type of relationship as 'is-a' does not capture the semantics of decomposition. For example, a Web service (process) in an IDEF0 model is not a business or IT process at a higher level diagram; it is an implementation of such process.

The following Table 2 summarises the above points.

Table 2. Ontological mappings for service governance.

Governance level	Purpose of governance	Example ontology	Possible type of governance automation systems
Service Portfolio	Capture of Organisation ontologies, goals, missions to drive identification/selection of suitable services	Enterprise ontology [10]	Organisational knowledge repositories to discover/mine services
SOA Architectural Compliance	Guide the design process, ensure services are correctly decomposed, obtain declarative specs of web services	The IDEF0 meta-model [5], Process ontologies, such as the Open Source Business Management Ontology [3]. Personal service ontologies, e.g. [7]	Integrated service development environments (IDEs) and ontology management systems.
Service Runtime	Check the runtime services for compliance with SLA rules and QoS constraints	Semantic service ontologies and Rules e.g. [9].	Service monitoring and execution environments, 'service bus'

6 Runtime Governance

A service oriented system, at runtime, is connected to its operational environment via multiple feedback mechanisms. For effective alignment with the environment, runtime e-services governance requires a mechanism for reasoning about continuous changing conditions that affect running services, and impact governance rules and constraints. Additionally, corrective actions may be required to bring the system back within the permitted operational boundaries.

For e-services governance within the continuous changing environmental conditions, it is, therefore, necessary to use a knowledge representation scheme capable of representing the structural as well as the behavioural semantics of e-services through governance rules that interpret feedback from the environment, identify discrepancies between expected and apparent states and resolve conflict.

To accomplish this, first governance rules need to describe the discrepancy between expected and received feedback and then link this with e-service variables defined in terms of an e-services ontology.

For this purpose, we propose a runtime governance environment that provides mechanisms for (a) alerting, in the light of feedback, regarding the behaviour of changing conditions at a given time, and (b) represent and use feedback in order to reason about the execution of governance rules to bring the system back to its permitted state.

A reasoning engine (RE) has been developed to address the issues above. It provides a rule based language capable of representing the interdependencies amongst service variables, expressed in terms of ontologies, and to support reasoning about continuous changing conditions within the scope of e-service governance.

Discrepancy Detection

The principal of conflict detection is based on detecting possible inconsistencies between the feedback and the expected value that indicate potential violation of governance rules. In order to carry out this approach, all governance rules in the system are searched for possible discrepancies. In other words, discrepancy detection is solely concerned with detecting inconsistencies between the values of e-service variables as defined in the governance rules and the values received from the environment. When a rule is found to be inconsistent, it is recorded as a failed rule. The discrepancy detection algorithm is defined as follows:

```
For each feedback value X received
do
    let V be the ontology variable whose X is its actual value
    for each Governance Rule R that contains V
    do
        if X == V.expected
            then
        R.status = "passed"
        else
        R.status = "failed"
        endif
    od
od
```

In the above algorithm, the attribute x refers to the value of an e-services variable, received as feedback from the environment. This could represent, for example, the roundtrip time between a service request and a service response. SLAs might be in place to constrain the permitted values for this response, in terms, for example, of average, maximum or minimum times etc. The expected values of these variables are described in the associated e-service ontology.

7 Discussion and Conclusions

SOA Governance has recently been proposed as an extension of IT Governance concept to e-services. SOA Governance models have been proposed by [4], [11], and others. In addition, the use of ontologies to describe services has been proposed both in the context of organisation level services [1], consumer level services [7], and Web services [9]. Ontologies have also been employed to assist with the specification of quality standards for services [8].

The approach presented in this paper emphasises the hierarchical nature of services governance. Because of their systemic aspect, service governance specification approaches need to allow us to move freely across levels of e-service definitions, from organisational to executable software, from service identification time to execution and monitoring time. Not all current SOA standards, however, have inherent support for hierarchical governance. In contrast, the IDEF0 modelling formalism we have adopted has an inherent capability to capture the semantics of governance in hierarchical service systems.

The key benefits of the proposed approach can be summarised as:

- Semi-automated support to propagate governance rules through the IDEF hierarchy.
- Achieved with the use of semi-automatic ontology mappings
- Consistency checking of Governance rules through the service levels
- Governance rules can be mapped to executable rules (e.g. in a scripting language, database triggers, etc)

Our continuing research aims to build a totally automated service governance environment that supports the concurrent specifications of e-services and governance policies, translates governance policies using ontologies, and enforces them at runtime using reasoning mechanisms (e.g. theorem provers and constraint satisfaction systems). We hope that this work will eventually produce self diagnosing and repairing e-service systems, making a step towards realising the vision of autonomic computing.

References

1. Baida, Z., Gordijn, J., Sæle, H., Akkermans, H., Morch, AZ. An Ontological Approach for Eliciting and Understanding Needs in e-Services. In Proc. CAISE 2005, Portugal, June 2005.
2. Basden, A. Summary of Dooyeweerd's Cosmonomic Philosophy. Available from http://www.dooy.salford.ac.uk/summary.html
3. BMO. The Open Source Business Management Ontology. Jenz & Partner. Available from http://www.bpiresearch.com/Resources/RE_OSSOnt/re_ossont.htm
4. Holley, K, Palistrant, J., Graham,S. Effective SOA governance. Available at http://www-05.ibm.com/e-business/uk/soa/pdf/effective_soa_governance.pdf
5. IFIP. Standard for INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0) Draft Federal Information Processing Standards Publication 183 1993 December 21. Available from http://www.idef.com/pdf/idef0.pdf

6. Matinlassi, M., Kalaoja, K. Requirements for Service Architecture Modeling. In Proc. WiSME@UML, 2002. Germany, 1/10/2002
7. Hung, JS., Liu, A. Using Personal Ontology in Evaluating Service Quality IEEE International Conference on Services Computing, 2007. SCC 2007. Volume , Issue , 9-13 July 2007 Page(s):332 – 339
8. Schmidt, R., Bartsch, C., Oberhauser, R. Ontology-based representation of compliance requirements for service processes. In Proc. Workshop SBPM 2007, Insbruck, April 2007.
9. Semantic Web services Ontology (SWSO) Version 1.0 Battle, S., Bernstein, A., Boley, H., Grosof, B., Gruninger, M., Hull, R., Kifer, M., Martin, D., McIlraith, S., McGuinness, D., Su, J., Tabet, S. (Ed.). Available from http://www.daml.org/services/swsf/1.0/swso/
10. Uschold, M., King, M., Moralee, S., Zorgios, Y. The Enterprise Ontology. The Knowledge Engineering Review, Vol. 13, Special Issue on Putting Ontologies to Use (eds. Mike Uschold and Austin Tate). 1998.
11. Wilkins, L. Governance in SOA. Available from URL:http://www.cbdiforum.com
12. Windley, PJ. SOA Governance: Rules of the Game. Infoworld.com. 01.23.06