Agent-Based Organisational Governance of Services

Frances Brazier[1], Virginia Dignum[1], Frank Dignum[2], Michael N. Huhns[3], Tim Lessner[4], Julian Padget[5], Thomas Quillinan[6], Munindar P. Singh[7], Christian Derksen[8]

[1] Delft University of Technology, The Netherlands
{f.m.brazier,m.v.dignum}@tudelft.nl
[2] Utrecht University, The Netherlands
dignum@cs.uu.nl
[3] University of South Carolina, USA
huhns@sc.edu
[4] University of the West of Scotland – Paisley, UK
timlessner@lesshome.net
[5] University of Bath, UK
jap@cs.bath.ac.uk
[6] Thales Nederland, The Netherlands
Thomas.Quillinan@d-cis.nl
[7] North Carolina State University, USA
singh@ncsu.edu
[8] Universität Duisburg-Essen, Germany
christian.derksen@icb.uni-due.de

Abstract. The objective of service-oriented computing (SOC) is to construct software applications out of appropriate services available and executing in place anywhere across the Web. To achieve this objective requires that techniques for discovering and engaging services be developed and used across the lifetime of the service-based applications. Succeeding with SOC in this broader sense presupposes that additional techniques be developed for ensuring desired quality of service metrics and service-level agreements. The crucial aspect of using services is thus their governance. In this paper, we propose a modelling framework that integrates organisational and coordination theories to achieve contextualised service governance. The approach allows for the development and analysis of dynamic, flexible, and robust service-oriented business applications.

Keywords: Institutions; Multi-agent systems; Organisations; Service engagements; Service enactment; Governance model

1 Introduction

The deployment of systems based on service-oriented architectures (SOA) is becoming widespread and successful in many application domains. Numerous commercial and civil organisations are actively engaged in converting their existing information systems or constructing new systems based on SOA principles [11]. However, the SOA-based systems being constructed are *static*, in that the services are known and fixed at design time, and their possible interactions are defined and characterised completely in advance. Services that can be *dynamically* discovered, configured, deployed, engaged, and maintained are envisioned [14], but are not yet successfully realized in practice [3]. The main obstacle to the expanded vision of services is that current service standards, which are necessary for the widespread usage of services, are unable to describe anything other than the simple syntax and formatting of service invocations; they are thus insufficient for characterising the rich usage and interactions required throughout the *lifetimes* of service-based applications, from discovery through maintenance [21].

In particular, service-oriented computing is intended to enable services to be discovered and enacted across enterprise boundaries [19, 17]. If an organisation bases its success on services provided by others, then it must be able to trust that the services will perform as promised, whenever needed. This entails having descriptions of the *behaviours* of the services, not just their functionality, so that their run-time interactions are predictable, observable, and controllable. Moreover, they must be predictable and controllable over a lifetime of interactions. Thus there is a need for what we call *service governance*.

The features of service governance lie well beyond what was originally envisioned for service-oriented architectures. These features include quality-of-service (QoS) and contracts, i.e., service-level agreements (SLAs) among the participants. Moreover, to make governance dynamically and autonomously configurable, the participants will need the ability to negotiate at run-time to establish the SLAs, to monitor compliance with them, and to take actions to maintain them. These are *software agent* capabilities. However, whereas the introduction of agents increases the flexibility of service interactions, it also introduces a new set of vulnerabilities, due to the autonomy—causing uncertainty—and asynchronous interactions—causing complexity—that characterise multi-agent systems.

The Web has been successful largely because its founding principles and protocols are simple and minimal. Also, when uncertainties arise, they are overcome by relatively simple approaches, such as indexing, ranking, and redundancy. None of these techniques has been exploitable for services. In addition, the simplicity of services applies only to their structure, and not to their function and behaviour, which have mostly been ignored in traditional service engineering.

Agents exacerbate the problems, while—perhaps surprisingly—also providing the only reasonable solutions to them. The autonomy of agent-based services makes them less predictable, but also enables them to recover from failure and to avoid deadlocks and livelocks, thereby making them more reliable. The ability of agent-based services to learn has the disadvantage of increasing their unpredictability, but has the advantage of increasing their robustness, because they can adapt to changing interaction environments. Their abilities to negotiate and reconcile semantic differences can enable them to re-establish connections and relationships with each other and ameliorate uncertain execution environments. The peer-to-peer interactions of agents can improve the efficiency of agent-based services, particularly when they are deployed in clouds. Finally, agents can exploit the redundancy provided by multiple alternative services.

In this paper, we present initial work towards a model for dynamic SOA, using agent-based technology, that provides different levels of abstraction for the specification of governance, expectations, and behaviour. In the next section we elaborate on the motivation for higher levels of abstraction to specify dynamic SOA. In section 4 the components of our governance model are discussed. The potential benefits of our governance model are illustrated in section 5. The paper finishes with conclusion's and discussion for future work in section 6.

2 Motivation

Our point of departure from previous characterizations of service-oriented architectures is the idea of a real-life service engagement [27, 5, 25]. A real-life service engagement inherently involves two or more largely autonomous and heterogeneous parties who exchange value with one another. To enact a real-life service engagement, participants naturally rely upon technical (Web or Grid) services. However, what distinguishes a real-life service engagement is that the interactions that constitute it are best understood in the business relationships among the parties, that is, on a higher level of abstraction than that of the services' specifications. In general, traditional operational ways of modelling and representing interactions are too low level to be appropriate. We suggest that such interactions should be captured as normative relationships realised within the scope of an institutional contract in which the participants carry out well-defined roles. In this section, we highlight the challenges and opportunities of governance through a series of scenarios of engagements of increasing complexity.

As an example, consider a simple service engagement corresponding to a two-party deal. Such an engagement might transpire when a user connects to Amazon to purchase an MP3 song, which the user can download directly from Amazon. (For simplicity, we ignore any additional parties needed to process payments.) In this situation, the conventional approach is for the user's application to invoke one or more of the Amazon Web services involved in searching for and purchasing the desired song. Notice that here the two parties find each other and interact in more or less standardised ways. There is a presumption that the application not only understands the data models of the Amazon services, but also understands that, at an appropriate time, the user becomes committed to paying for the selected song and that Amazon becomes committed to providing the media for the song.

In contrast, we understand the evolving relationships among the participants as central to the above engagement. If either party fails to perform according to its commitments, we consider such failure to be a violation, regardless of the low-level means through which the participants happen to interact and the operational details of the order in which they communicate. Moreover, it makes a huge difference at which point possible failures of the interaction happen. If the technical interaction fails after the search but before downloading and payment, no harm is done. The transaction can either be canceled or restarted. However, if there occurs a failure between the downloading and the payment (in whatever order they take place) we cannot just cancel the transaction. This (technical) failure has larger repercussions. Moreover, recognising that a real-life service engagement involves autonomous parties means that (besides possible technical failures) there is no inherent guarantee that each party will be well behaved.

For these reasons, and following common practice, it is convenient or even essential that we model the organisational or social scope within which the engagement takes place. The scope provides a facility to monitor the interactions, record reputations, assist in matchmaking, and ensure compliance. Unless the scope has legal powers, it might only be able to ensure compliance based on threats of censure, removal of a malfeasant participant, or escalation to a higher (such as a legal) authority.

Moreover, taking seriously the above-mentioned point about capturing normative relationships as opposed to operational details, we grant first-class status to the scopes as institutions. In this sense, an institution is an organisation that has an identity of its own and to which different parties must belong in order to interact with one another. An institution could be an existing social entity (such as a university or the State of North Carolina in the US). Or, it could be an entity that we construct for a certain family of engagements, in which case we might term it a virtual organisation. The distinction is largely irrelevant for our present purposes.

We can understand an institution as including itself as a distinguished member along with the parties carrying out a service engagement. Further, we specify an institution in terms of the normative relationships it imposes among its members. Some of the normative relationships arise between the transacting parties and some between them and the institution itself—indeed this is why we model the institution as a distinguished member. To this end, it is important to define the *roles* of an institution as descriptions of how members of different types are expected to behave.

To return to the Amazon example, we can imagine the customer and Amazon each participating in the *Commerce* institution, which can be defined as being subject to GS1 Standards and other rules of (commercial) encounter [12]. The *Commerce* institution in this sense is what one implicitly finds when conducting commerce within a legal jurisdiction. As members of the institution, each party is permitted to buy and sell items it owns, but prohibited from trading illegal objects (such as drugs). Violations would expose each party to sanctions.

The governance of a service engagement is how it is administered, especially when carried out by the participants [27]. The normative relationships placed within an institution as scope provide a natural way to achieve governance. In particular, by using high-level specifications, we can characterise the requirements of the stakeholders precisely without having them over-constrained by operational details. In the above case, the initialisation and enactment of the engagement is constrained by the institutional scope. The parties can decide what to trade and can escalate the interaction to the scope (that is, by complaining), if the other party does not keep its end of the deal.

The situation becomes more interesting in engagements where the institutions are not directly the legal system, but something more flexible. To this end, as our second example, one can think of a customer purchasing goods not from Amazon, but from one of the sellers on Amazon Marketplace. Here the buyers and sellers must obtain accounts on Amazon and Amazon serves as the institutional scope. In obtaining an account, each party enters into a contract with Amazon that specifies the rules of encounter. In an actual business transaction, a buyer and seller would meet through Amazon, negotiate a price (which might be realised through a fixed offer or an auction), commit, and discharge their respective commitments to enact the engagement. Governance is richer here because of the extra layer of the Amazon scope.

Notice that the enactment of such engagements would naturally include the parties stepping out of the scope physically, even though they remain bound to it logically. Specifically, the parties will use external means, such as payment and shipping agencies, which are not bound to the Amazon rules of encounter and do not exist within the Amazon scope. However, failure by such external means can cause violations of normative relationships within the Amazon scope, and the participants may suffer the consequences as a result.

Finally, service engagements also arise, perhaps more prominently, in business-to-business settings. For our third example, we consider interactions between enterprises, such as between an automobile manufacturer and a parts supplier. Each party is an organisation in its own right. We can imagine that they carry out their trade in the generic *Commerce* institution that is described above or through a more confined institution such as a Parts Exchange (analogue of Amazon Marketplace) or even something less formal, such as a Japanese keiretsu. Agents playing suitable roles in each enterprise viewed as an organisation (that is, empowered with signatory authority) are necessary to instantiate the contract. However, the enactment of the contract requires the participation of sub-organisations and their representative agents lower down the respective corporate hierarchies. We can imagine this enactment by participation as a combination of delegation and assignments of the high-level normative relationship as well [25, 27]. The engagement proceeds smoothly if such coordination succeeds, but may fail otherwise. In general, any of the parties may escalate what was delegated or assigned to them, generally to the agents who originated the delegation or assignment. In other words, each party would complain to its manager or designated agent in the super-organisation. The peers may be able to renegotiate details such as delivery times and trucks (or not— depending on their powers and authorisations within their respective organisations). If the engagement cannot recover within the organisations, there may be an escalation to the scope—analogous to an escalation to the Amazon Marketplace if the seller does not send the buyer the ordered goods. The above thus illustrates governance at the level of performance, renegotiation, and escalation.

The foregoing discussion describes service engagements and how their governance is naturally viewed in terms of normative relationships arising within suitable institutions. The same ideas can be applied in finer granularity in terms of agreements about specific qualities of service. By formulating governance in these terms, we can show how service engagements can be carried out and administered in a manner that respects the details of technical services without being bogged down in their details.

It should be noticed that norms in institutional or organisational specifications tend to abstract from the concrete events and situations that the norm is supposed to cover. The norms of institutions are intentionally specified at a high level of abstraction to range over many different situations and to require little maintenance over time. Whereas this abstraction creates increased stability over time and flexibility of application for the norms, it also creates a problem when using norms, because the abstract concepts in norms need to be related to the concrete events and concepts that occur on the technical service level of the system. [1] has shown how links can be made between norms on the different levels (using a practical account of the counts-as concept), while each level can concentrate on the issues important for that level.

3 Related Work

Advances in service-oriented architectures and successes in initial service-oriented applications have spurred new and larger development efforts and encouraged optimistic predictions of future Internet-wide applications of services. For example, the ALIVE Project has envisioned 1000's of services working in concert to produce desired results, even when faced with requirments that change and services that fail at run-time [2]. However, although small numbers of services have been deployed successfully in closed environments, the promise of large numbers of interchangeable or easily replaceable services in open environments has not been realized. As described above, the problem is in how the services are governed.

The motivation for a provider to develop and deploy a service is typically economic: the service might be used widely and become a revenue stream for the provider. It thus behooves the provider to ensure that the service is operable, behaves correctly, and maintains its commitments for functionality quality-of-service. Basically, the provider wants to keep its clients happy, and it can do this by engaging in dialogs with the clients when there are problems that arise and need to be resolved. This requires action, so ordinarily passive software services must behave like active software agents.

Following the development of individual agents by early researchers in multi-agent systems, attention was focused on large collections of interacting agents, which required some form of organisation to operate efficiently. Concepts from human societies were adopted and adapted for use by the agents, with the concepts covering the behaviours of the agents (e.g., norms), the interactions among the agents (e.g., commitments and protocols), the environment in which the agents operated (e.g., observable and deterministic), and the relationships among the agents (e.g., hierarchical). This is covered well in [].

There have been numerous approaches to SOA governance. The approaches might operate at eihter design-time or run-time. Eight design patterns for governance have been identified for design-time governance [10]. A recent book focuses especially on run-time governance, but for closed applications [4].

The paper by Dignum et al. [6] delineates the problems that are likely to be encountered when services are deployed and used widely. It proposes that the ALIVE architecture [33] be used to address the problems, where the ALIVE architecture extends an abstraction layer consisting of individual services and their semantics with coordination and organisation layers. The coordination layer encapsulates each coordinating service in an agent that enables these services to be active participants in a workflow. The workflow might be formed dynamically at run-time and may extend across enterprise boundaries. The coordinating agents can invoke simple services from the service layer to enact the workflows and can provide monitoring services. The organisation layer provides a social context for the other two layers, where the goals, rules, norms, and results of a system can be specified to govern the interaction of its components. More concretely, an organisation is a set of entities, the roles they play, and their interactions [1]. The definition of the context provided by the organization layer is the major innovation of the ALIVE architecture.

The ALIVE Project is developing a tool suite to enable its architectural concedpts to be realized and used. For an initial stage of an application, it provides a tool for modeling the organisation layer and its components. It also provides a tool for designing the workflows and their coordinating agents. A service-design tool is used to specify the semantics of services and store the resultant service descriptions in a repository, where a matchmaking tool can be used to find and engage them.

The ALIVE architecture is demonstrated via a case study in crisis management.

4 Governance Model

Based on the challenges identified in the previous sections, we describe a governance model for virtual organisations that comprises three levels (inspired by [6]): (1) organisation level, (2) agent level, and (3) service level. We then extend this model by specifying declaratively the knowledge and processes involved and elaborate it further by considering the structural, functional, and behavioural aspects of the knowledge. We then note how the model changes according to whether it is used at design time or run time.

The Organisation Level provides a social context for the service architecture, specifying the actors and institutions involved, their objectives, requirements and dependency relations, and the organisational rules that govern interaction [7]. Organisations describe real-life engagements, including their context, expectations, and norms. The relationships between agents are defined by the organisations to which they belong, but agents are led by their own reasoning abilities, desires, and beliefs. Agents activate services to achieve their goals. As such, the Agent Level provides the means to specify, at a high level, the patterns of interaction among services, using a variety of powerful coordination techniques from recent agent research [15]. The agents are intended to be organisationally-aware, meaning they are aware of overall system objectives and context, and are able to manage task allocation and workflow generation and agreement.

Services, or compositions of services, are encapsulated in the agents that make them available to others. The Service Level extends existing service models, to make components aware of their social context and of the rules of engagement with other components and services, by means of Semantic Web technologies. Increased semantics is particularly useful when highly dynamic and frequently changing services are present, as the meta-information in each service description (stored in a service directory) facilitates tasks such as finding substitute services, either via a matchmaker or manually, when the original service fails. This multi-level meta-architecture can be seen as a service-oriented middleware supporting the combination, reorganisation, and adaptation of services at both design-time and run-time [32, 23]. These activities follow organisational patterns and adopt coordination techniques. Table 1 provides an overview of the model.

At each of the three levels (1) the *knowledge* and (2) the *processes* involved are distinguished. The knowledge dimension identifies the ontology used to construct the system. Knowledge about *structure* (**S**), *function* (**F**), and *behaviour* (**B**) in the system design is often scattered across the application domain. This ontology provides the fundamental concepts for capturing the target domain and a common vocabulary for description and integration of the different levels. Processes are described at different abstraction levels, with different objectives (e.g., management decisions at the organisa-

	Knowledge (ontology)	**Process**
(Virtual) Organisation (Representative)	**Structure:** control structure, roles, values, type **Function:** norms, purpose **Behaviour:** quality of ethos	Business model defining normative relationships
Agent (Owner)	**Structure:** communication, decision-making strategies, level of autonomy **Function:** goals, mores/values, BDI **Behaviour:** Quality of Character	Coordination model
Service (Provider)	**Structure:** data types, syntax, interfaces **Function:** declarative semantic description **Behaviour:** Actual QoS	Enactment

Table 1. Model Overview

tional level and system implementation at the service level). The ontology relates these processes, such that decisions and requirements at one level can be related and, when appropriate, propagated to the other levels.

In order to link a system to the actors that enable, enforce, or own it, each level is associated with a different category of actors. Each (virtual) organisation has a *Representative*. Each organisation has its own characteristics, structure of control, roles, and responsibilities, i.e., its structure. Organisations have their own norms and purpose: goals and ethical function. The Quality of Ethos determines the way organisations are perceived. Agents have *Owners*. Agents have a level of autonomy, communicate with other agents, and make individual or collective decisions. They have individual goals, mores and values, beliefs, desires, and intentions. Quality of Character defines the way they are perceived, and determines their reputation. Services are provided by *Service Providers*. Agents activate services by using the syntax, data types, and interfaces published. Services are often chosen on the basis of their declarative semantic descriptions. SLAs define the expected quality of service and the conditions. A service is best described by the actual quality of service it provides.

In the next sections we describe in more detail each of the levels and their role in the governance model.

4.1 Organisation

One of the main reasons for creating organisations is to provide the infrastructure for coordination that fosters the achievement of global goals. Organisational structure has essentially two objectives [9]. First, it facilitates the flow of information within the organisation in order to reduce the uncertainty of decision making. Second, it integrates behaviour across the parts of the organisation so that it is coordinated.

The design of organisational structures and processes determines (1) the allocation of resources and agents to specified tasks or purposes, and (2) the coordination of these resources to achieve organisational goals. Both in human enterprises and in multi-agent systems the concept of structure is central to the design and analysis of organisations [8].

Williamson argues that the transaction costs are determinant for the organisational model [35]. Transaction costs will increase when the unpredictability and uncertainty of events increases, or when transactions require specific investments, or when the risk of opportunistic behaviour of partners is high. When transaction costs are high, societies tend to choose hierarchical models in order to control the transaction process. If transaction costs are low, that is, are straightforward and non- repetitive and require no transaction-specific investments, then a market is the optimal choice. Powell introduces networks as another possible coordination model [22]. Networks stress the interdependence between different organisational actors and pay a lot of attention to the development and maintenance of (communicative) relationships, and the definition of rules and norms of conduct within the network. At the same time, actors within networks are independent, have their own interests, and can be allied to different networks. That is, different business models are based on different environment strategies and define normative relationships:

- *Strict hierarchical organisation* — well-structured, with well-defined delegation of tasks, responsibilities, authority, reporting, monitoring, and supervision and control.
- *Networks* — groups of organisations that together agree to collaborate, collectively negotiate how to delegate tasks and responsibilities, how to monitor task progress, and how to regulate their collaboration. Trust plays an important role.
- *Completely distributed open market* — competitive, full autonomy of individual organisations, where cooperation depends solely on perception of mutual benefit.

Each of these types of organisations comes with a set of predefined patterns and organisational roles. Thus, depending on the context in which a service-oriented application is developed, a certain organisation type can be chosen that fits best. For instance, the automobile manufacturing enterprise, which is used in the case study below, needs a network organisation, because manufacturing partnerships range over sequences of transactions and trust about the timely delivery of parts is important. The Amazon marketplace is a good example of a market organisation, with its established norms and roles.

4.2 Agents

Agents are the decision-making entities within an organisation: they activate services to achieve their goals. This means service invocation is goal directed. This enables finding alternative services and replanning when services fail to comply to SLAs. In that case, alternatives are sought that achieve the same goal, but might differ in implementation details.

Agents can also act as representatives for an organisation. This feature allows for a hierarchical specification of an organisation in terms of divisions, departments, etc. The example in the next section illustrates this point, where an automobile factory is part of and represents a larger automobile manufacturer.

In complex organisations, coordination between agents is mandatory to manage dependencies among their activities. Different types of coordination can be distinguished: (1) functional coordination, (2) temporal coordination, and (3) information coordination.

Functional coordination refers to the delegation of control within an organisation: ranging from fully distributed (within which each agent fends for itself in a fully networked environment) to fully controlled in a hierarchical structure (within which agents only perform the tasks they have been delegated, upon request, and only interact with agents with whom interaction has been requested by a higher-level agent). Mediated structures in which aspects of both extremes may be combined are often encountered in practice.

Temporal coordination refers to the timing of a process. As agents are autonomous, they each have their own sense of time, that is, their own clock. Coordination requires the temporal aspects of interaction to be considered during design (even though, in fact, interaction in distributed environments is, by definition, asynchronous). Dependencies need to be defined and incorporated in interaction patterns known to the agents.

Information coordination refers to the information agents need to have to be able to reach their goals. In situations in which agents need to interact, information dependencies are crucial to their individual ability to perform. If information is to be shared, there must be a means by which this is achieved: e.g., shared memory, broadcasting, multi-casting, and message passing.

Each agent's main purpose in a complex organisation is to participate in the different types of coordination at run-time. So, when unexpected events happen, the agents can adapt to the resultant new situation by rearranging and coordinating the changes. This is a significant advantage over service choreography where coordination is arranged at design-time.

4.3 Service Enactment

Agents enact services to reach their goals. Service interactions might be complex, so that agents might need to orchestrate their behaviour. They may need to adapt and engage a new agent (whoswe selection would be based on the design of the system) if something goes wrong. To this purpose an agent must be able to schedule service-enactment, monitor and influence service performance, re-orchestrate a complex service if needed, and influence the choreography. An agent must be capable of detecting malfunctioning behaviours and to adapt appropriately.

To simplify the tasks of the agents, services could be clustered according to their functionality and agents could then manage and use the clusters to help locate possible services to satisfy requests from users and developers. A procedure such as unit testing could then be used as a behavioural query tool to test candidate services and select ones that have the desired behaviour [13]. A negotiation between the service requestor and providers could then ensue to establish an agreed upon QoS and formalise an agreement. The requestors and providers would commit to honour the resultant con-

Stakeholders: Organisation	Knowledge (ontology)	Process
– Suppliers – Customers • Dealers • Individuals – Shareholders – Manufacturers – Sellers	**Structure:** Board of Directors; Shareholders meeting. Negotiated agreements **Function:** Produce automobiles suitable for market. **Behaviour:** Profit; Cost Control for customers; Safety; Timeliness; "Greenness".	Required to make a profit. Cooperate / Negotiate for resources and sales. Timely delivery of cars.

Table 2. Stakeholders: organisation view

tracts. The above scenario presupposes agent abilities that are beyond those of regular services.

5 Illustrative Scenario

In this section, an automobile manufacturing company is modelled using the structure outlined in Section 4. In this example, two organisations are described within the company: the corporate stakeholders and the manufacturing business. The stakeholders are responsible for the overall corporate direction and manage the business. The manufacturing business is one part of the business responsible for building specific automobiles. These aspects are described separately.

5.1 Stakeholders

Modelling the organisation of the stakeholders entails determining the actors, including customers, suppliers, manufacturers, shareholders, and sellers. These actors all have a stake in the running of the business. The structure is formal—a board of directors as well as the requirements to hold regular shareholder meetings. Customers and suppliers have formal agreements that specify their relationships. The behaviour of the business includes requirements that can be in conflict (such as cost control, profit, and "greenness." In order to be successful, an appropriate balance must be attained. The above is summarised in Table 2.

A number of aspects of the organisation can be specified as agents. In this example, only a seller agent is modelled in Table 3. For this agent, the structure relates to the interactions between the corporate stakeholders, the customers, and the manufacturing aspect of the business. In order to achieve the agent's goals, it uses a number of services.

The ordering service (summarised in Table 4) is enacted by the seller agent, placing orders from customers with the manufacturing business. This requires a specific interface to the service, specifying the type of automobile, the customer identifier, as well as preferences such as colour and interior. A service-level agreement is used to define the acceptable quality-of-service guarantees that have been negotiated. These include absolute dates for manufacturing as well as penalties for non-compliance.

Stakeholders: Agents	Knowledge (ontology)	Process
Seller Agent (Stakeholders)	**Structure:** One-to-one negotiation with customers. Not allowed to sell below cost. Limited ability to discount. **Function:** Make a profit. Sell as many automobiles as can be produced. **Behaviour:** Must be seen as honest and dependable. A good reputation helps sales.	Coordinates between manufacturing and customers to determine the price point and the number of automobiles that can be produced. Orders are sent to the manufacturing sub-organisation.

Table 3. Stakeholders: agent view

Stakeholders: Services	Knowledge (ontology)	Process
Ordering Service (Seller)	**Structure:** order(Order ID, Car Type, Preferences, Due date, Customer ID) **Function:** Orders an automobile of a specified type from the manufacturer and specifies the desired due date. **Behaviour:** SLA defines the acceptable time allocated to manufacturing and delivering automobiles. This is different from the due date as it may specify penalties for non-compliance.	Enactment of the service.

Table 4. Stakeholders: ordering service view

An additional requirement is for a return service (depicted in Table 5) that describes how automobiles are returned to the manufacturer if they are not accepted, which might be because they are found to be defective. Defective automobiles may be physically defective or automobiles that are no longer required due to customers changing their mind. SLAs define the service parameters for accepting a return, as well as the possible fees for returning automobiles that are no longer required yet are functionally correct.

5.2 Manufacturer

The manufacturer is one of the stakeholders, as well as an operating agent in the previous model. Examining the structure of the manufacturer supports creating a more specific model (see Table 6). In this case, the organisation is made up of suppliers,

Stakeholders: Services	**Knowledge (ontology)**	**Process**
Return Service (Seller)	**Structure:** return(Car Type, Order ID, Customer ID) **Function:** Return an automobile to the manufacturer. An automobile might be returned because it is defective or because its order was cancelled or changed. **Behaviour:** SLA defines how long a refund should take as well as the "restocking fee" that applies when an order is cancelled for non-functional reasons.	Enactment

Table 5. Stakeholders: return service view

Manufacturer: Organisation	**Knowledge (ontology)**	**Process**
– Suppliers – Shareholders – Board of Directors – Workers	**Structure:** Building cars. **Function:** Produce cars for sale. **Behaviour:** Fulfill requirements of, e.g., safety, efficiency, suitability, and cost control.	Required to efficiently produce automobiles using supplies to specifications. Sufficient parts should be supplied, without maintaining a large stock.

Table 6. Manufacturer: organisation view

the manufacturing workers, and the controlling entities: the board of directors and the shareholders. The structure is in place at this point to actually manufacture automobiles for the seller agents to perform their tasks to sell automobiles. The requirements here are to produce the automobiles safely and efficiently while managing the costs associated with their production.

The manufacturer has an assembly agent (described in Table 7) that maintains both the supplies and the order schedule so that automobiles are not delayed, causing penalties. This agent determines the manufacturing schedule so that urgent orders are scheduled quickly and parts are ordered from suppliers to arrive in time for assembly.

One service that the manufacturer requires is a service to manage supplies of automobile parts (depicted in Table 8). This service provides stock management as well as automated ordering of parts based on existing service-level agreements. This service is used by the assembly agent to ensure that assembly is successful.

Manufacturer: Agents	Knowledge (ontology)	Process
Assembly Agent (Manufacturer)	**Structure:** Determine when to order supplies; Determine the schedule to start building received orders; Build automobiles quickly and cheaply. **Function:** Build automobiles efficiently: Ensure stocks of supplies are appropriate for upcoming orders. Prevent penalties from occurring. **Behaviour:** Efficient, dependable, safe.	Coordinates between suppliers and orders appropriate parts. Coordinates construction schedule to ensure order deadlines are met.

Table 7. Manufacturer: agent view

Manufacturers: Services	Knowledge (ontology)	Process
Parts Service (Assembly Agent)	**Structure:** orderSupplies(Part ID, Quantity, Date) **Function:** Order parts from suppliers. This specifies the date when the part is required as well as the functional details of quantity and part identifier. **Behaviour:** SLA specifies the deadlines and might describe the price depending on how quickly the part is required.	Enactment

Table 8. Manufacturer: assembly service view

As can be seen from the above examples, the model described in Section 4 provides the ability to capture the details of a business at the three key levels of abstraction, with emphasis on the behavioural aspect, which is the foundation for governance.

6 Discussion and Conclusion

Technology and economics are together driving the development of open systems: one pushing and the other pulling, but as yet the realisation of open systems remains just out of reach for a range of complicated and interrelated reasons. The purpose of this section is to lay out our perspective on these questions, highlighting the (often complementary) research that is taking place in different areas of computer science and that we suggest should be brought together to close the gap between the current state and the desired state of effective delivery of open systems. As explained above, our paper is centred on the adoption of a governance perspective.

For the last two decades, CORBA, and its like, have offered the primary approach to systems integration—whereby we mean joining legacy systems with new ones and maintaining current systems on an on-going basis. While this has provided a means to connect, but decouple, a variety of software components, the emphasis has typically been on delivery *within* an organisation. During this period, web services have appeared and there has been migration to this technology, but more as a substitute, again within an organisation, rather than across organisations as envisioned originally. Some of the factors that discourage uptake, as discussed at greater length above, include: operational rather than functional service descriptions, difficulty in monitoring service provisioning, problems in locating and handling faults, and determining responsibility in the case of incomplete delivery of a service.

The scientific computing revolution has centred on the development and evolution of the grid, which has adopted web services despite the shortcomings noted above, because the research community is relatively happy to trade resources (you can use my computer if I can use yours) and is more tolerant of, and less litigious, when faced with failure of provision. Additionally, the demands of capability computing, characterised by long-running programs and complicated workflows of several such programs, have driven the development of distributed workflow engines, e.g., Taverna [20], YAWL [31], WS-BPEL [18], and monitoring systems, with the objectives of (i) raising the programming task to one of composing services using a variety of familiar procedural constructs and (ii) handling service failure graciously so that it need not result in workflow failure [26].

Service-level agreements are playing an increasing role in grid computing, not just as a way for service consumers to state their requirements, but both as a basis for negotiation (Mobach et al., 2006) between consumer and provider and as a way of scheduling [24, 34] the best use of resources by reconciling the conflicting demands of throughput and response time. It seems likely that SLAs have an important part to perform in capacity computing (clouds) in helping to establish the practice of electronic contracts for intangible goods at the same time as refining the language and instruments of SLAs through experience. A particular challenge here is that it is rare that just one SLA will suffice: much more likely are *hierarchies* of SLAs, where constituent tasks are governed by further SLAs, but the primary contractor is not, and does not wish to be, aware of such details. Some aspects of these issues are considered by Haq et al. [29, 28] who examine business value networks and build on WS-Agreement [16] and Unger et al. [30] who focus on business process out-sourcing and utilise WS-Policy[1]. Some factors discouraging uptake of SLAs include the relative immaturity—by business standards—of the tools and the lack of a proper legal understanding of and status for SLAs, but there is growing interest, as well as a realisation of the necessity of such approaches.

Workflows began as compositions of specific services, but there are two factors encouraging abstraction: (i) the desire for re-usability and (ii) the gradual uptake of

[1] Web Services Policy 1.2 - Framework (WS-Policy), http://www.w3.org/Submission/WS-Policy/. Retrieved 20100307.

open systems, both of which force a shift from exactly *what* service to use to specifying *how* the service shall function. This can be brought about through semantic service description languages such as OWL-S, consequently monitoring of workflow progress can be expressed in application terms and service failures might be resolved by finding functional substitutes. But different organisations will inevitably have different views on what information matters about the progress of a workflow, and likewise different policies with respect to what constitutes an adequate substitute.

Software agent technology has now matured sufficiently that it is accepted as a way of thinking about systems construction that works with other components, rather than a kind of hegemony that seeks to impose a one-size-fits-all solution. Of particular potential benefit from this domain is the research on automated negotiation—which is already feeding into the refinement of WS-Agreement—and argumentation, distributed resource allocation and aggregation techniques and, perhaps most appropriately given the case study in Section 5, the development of formal approaches to organisational modelling. These last offer the opportunity to construct machine-processable policies that capture high-level organisational intentions, yet whose influence reaches down to individual decisions, such as those highlighted in the preceding paragraph.

Looking back at the above, we identify (i) dynamic behaviour, (ii) formalisation of business roles and rules, (iii) response to change (over short and long term), and (iv) formalisation of agreements (in the physical and the virtual world) as the key challenges to be met to achieve the next level of aspirations in automated service provisioning. We have gone beyond architectural approaches, such as ALIVE, that emphasize roles and dependencies, which corresponds to our structural and functional aspects, but do not consider the behavioural aspects of knowledge explicitly. The behavioural aspects mesh naturally with the concept of commitments and, thus, to governance. Moreover, we believe it is clear from this necessarily partial view of a broad range of research that good foundations exist on which to build the next steps in delivering service-oriented applications.

We also believe it is important to delineate explicitly

- Applications that are developed in open environments and applications that are developed within closed environments (traditional SOA applications)
- Applications that execute in open environments and applications that execute within closed environments (traditional SOA applications)
- Knowledge and processes
- Structure (know-what), function (know-how), and behaviour (know-when)
- Organisations, agents, and services

In conclusion, what we have offered here is a broad assessment of the state of service-oriented applications, in which we identify a need to address the consequences of a changing environment in which such systems may be deployed. We propose that the combination of software agents and organisational modelling are well-suited to the task of providing an agile management layer whose function is directed by the dynamic interpretation of formal models of governance. In so doing, we seek to build on a broad range of research in services, agents, workflows, the Semantic Web, and grid computing, each of which brings its strengths to a complex, layered, architecture.

Acknowledgements: This work is the result of many fruitful discussions during the Dagstuhl workshop on "Service-Oriented Architecture and (Multi-)Agent Systems Technology" held in January 2010. We thank the organisers and all the participants for their contributions.

References

1. Huib Aldewereld, Sergio Alvarez-Napagao, Frank Dignum, and Javier Vazquez-Salceda. Making norms concrete. In Wiebe van der Hoek, Gal Kaminka, Yves Lesprance, Michael Luck, and Sandip Sen, editors, *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 807–814, 2010.
2. Huib Aldewereld, Julian Padget, Wamberto Vasconcelos, Javier Vazquez-Salceda, Paul Sergeant, and Athanasios Staikopoulos. Adaptable, organization-aware service-oriented computing. *IEEE Intelligent Systems*, July/August:80–84, 2010.
3. Alex E. Bell. From the front lines: Doa with soa. *Communications of the ACM*, 51(10):27–28, October 2008.
4. Stephen Bennett, Thomas Erl, Clive Gee, Robert Laird, Anne Thomas Manes, Robert Schneider, Leo Shuster, Andre Tost, and Chris Venable. *SOA Governance: Governing Shared Services On-Premise and in the Cloud.* Prentice Hall/PearsonPTR, 2011.
5. Nirmit Desai, Amit K. Chopra, and Munindar P. Singh. Amoeba: A methodology for modeling and evolution of cross-organizational business processes. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 19(2):1–45, 2009.
6. Frank Dignum, Virginia Dignum, Julian Padget, and Javier Vazquez-Salceda. Organizing web services to develop dynamic, flexible, distributed systems. In *Proceedings of 11th International Conference on Information Integration and Web-based Applications & Services*, pages 225–234. ACM, 2009.
7. Virginia Dignum. *A Model for Organizational Interaction: based on Agents, founded in Logic.* SIKS Dissertation Series 2004-1. Utrecht University, 2004. PhD Thesis.
8. Virginia Dignum, Frank Dignum, and Liz Sonenberg. Design and analysis of organizational adaptation. In L. Yilmaz and T. Ören, editors, *Agent-Directed Simulation and Systems Engineering*, pages 239–269. Wiley, 2009.
9. Robert Duncan. What is the right organizational structure: Decision tree analysis provides the answer. *Organizational Dynamics*, Winter:59–80, 1979.
10. Thomas Erl. *SOA Design Patterns.* Prentice Hall/PearsonPTR, 2009.
11. The Open Group. Soa case studies, 2008.
12. GS1. Gs1 2011 overview. Technical report, GS1 US, 2011.
13. Rosa Laura Zavala Gutierrez, Benito Mendoza, and Michael N. Huhns. Behavioral queries for service selection: An agile approach to soc. In *Proceedings of the IEEE International Conference on Web Services*, pages 1152–1153, July 2007.
14. Michael N. Huhns, Munindar P. Singh, Mark Burstein, Keith Decker, Ed Durfee, Tim Finin, Les Gasser, Hrishikesh Goradia, Nick Jennings, Kiran Lakkaraju, Hideyuki Nakashima, Van Parunak, Jeffrey S. Rosenschein, Alicia Ruvinsky, Gita Sukthankar, Samarth Swarup, Katia Sycara, Milind Tambe, Tom Wagner, and Laura Zavala. Research directions for service-oriented multiagent systems. *IEEE Internet Computing*, 9:65–70, 2005.
15. Mihhail Matskin, Peep Küngas, Jinghai Rao, Jennifer Sampson, and Sobah Abbas Petersen. Enabling web services composition with software agents. In *Proc. 9th IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA 2005)*, pages 93–98. ACTA Press, 2005.

16. David G. A. Mobach, Benno G. Overeinder, and Frances M.T. Brazier. A ws-agreement based resource negotiation framework for mobile agents. *Scalable Computing Practice and Experience*, 7(1):23–36, 2006.
17. OASIS. Reference model for service oriented architecture 1.0. http://docs.oasis-open.org/soa-rm/v1.0/, October 2006.
18. OASIS. Web services business process execution language (ws-bpel). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel, April 2007. Retrieved 20100307.
19. OASIS. Reference architecture for service oriented architecture version 1.0. http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf, April 2008.
20. Thomas M. Oinn, R. Mark Greenwood, Matthew Addis, M. Nedim Alpdemir, Justin Ferris, Kevin Glover, Carole A. Goble, Antoon Goderis, Duncan Hull, Darren Marvin, Peter Li, Phillip W. Lord, Matthew R. Pocock, Martin Senger, Robert Stevens, Anil Wipat, and Chris Wroe. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience*, 18(10):1067–1100, 2006.
21. Michael P. Papazoglou and Willem-Jan Van Den Heuvel. Business process development life cycle methodology. *Communications of the ACM*, 50(10):79–85, October 2007.
22. Walter W. Powell. Neither market nor hierarchy: Network forms of organisation. *Research in Organisational Behaviour*, 12:295–336, 1990.
23. Thomas Quillinan, Frances Brazier, Huib Aldewereld, Virginia Dignum, Frank Dignum, Loris Penserini, and Niek Wijngaards. Developing agent-based organizational models for crisis management. In *Proc. 8th Joint Conference on Autonomous and Multi-Agent Systems (AAMAS 2009)*, 2009.
24. Rizos Sakellariou and Viktor Yarmolenko. Job scheduling on the grid: Towards sla-based scheduling. In L. Grandinetti, editor, *High Performance Computing and Grids in Action*, volume 16 of *Advances in Parallel Computing*, pages 207–222. IOS Press, 2008.
25. Munindar P. Singh, Amit K. Chopra, and Nirmit Desai. Commitment-based service-oriented architecture. *Computer*, 42:72–79, 2009.
26. Rafael Tolosana-Calasanz, José A. Bañares, Omer F. Rana, Pedro Álvarez, Joaquin Ezpeleta, and Andreas Hoheisel. Adaptive exception handling for scientific workflows. *Concurrency and Computation: Practice and Experience*, 22(5):617–642, 2010.
27. Yathiraj B. Udupi and Munindar P. Singh. Governance of cross-organizational service agreements: A policy-based approach. *Services Computing, IEEE International Conference on*, 0:36–43, 2007.
28. Irfan ul Haq, Altaf Ahmad Huqqani, and Erich Schikuta. Aggregating hierarchical service level agreements in business value networks. In Umeshwar Dayal, Johann Eder, Jana Koehler, and Hajo A. Reijers, editors, *BPM*, volume 5701 of *Lecture Notes in Computer Science*, pages 176–192. Springer, 2009.
29. Irfan ul Haq, A. Paschke, Erich Schikuta, and H. Boley. Rule-based workflow validation of hierarchical service level agreements. In *Workshops at the Grid and Pervasive Computing Conference*, pages 96–103, 2009.
30. Tobias Unger, Frank Leymann, Stephanie Mauchart, and Thorsten Scheibler. Aggregation of service level agreements in the context of business processes. In *EDOC*, pages 43–52. IEEE Computer Society, 2008.
31. Wil M. P. van der Aalst and Arthur H. M. ter Hofstede. Yawl: yet another workflow language. *Inf. Syst.*, 30(4):245–275, 2005.
32. Javier Vazquez-Salceda, Luigi Ceccaroni, Frank Dignum, Wamberto Vasconcelos, Julian Padget, Siobhan Clarke, Paul Sergeant, and Kees Nieuwenhuis. Combining organisational and coordination theory with model driven approaches to develop dynamic, flexible, distributed business systems. In *Digital Business*, volume 21 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 175–184. Springer, 2010.

33. Javier Vazquez-Salceda, Wamberto W. Vasconcelos, Julian Padget, Frank Dignum, Siobhan Clarke, and M. Palau Roig. Alive: An agent-based framework for dynamic and robust service-oriented applications. In Wiebe van der Hoek, Gal Kaminka, Yves Lesprance, Michael Luck, and Sandip Sen, editors, *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 1637–1638, 2010.
34. Philipp Wieder, Oliver Wäldrich, and Wolfgang Ziegler. Advanced techniques for scheduling, reservation, and access management for remote laboratories. In *e-Science*, page 128. IEEE Computer Society, 2006.
35. Oliver E. Williamson. *Markets and Hierarchies: Analysis and Antitrust Implications*. Free Press, 1976. ISBN 13: 9780029353608.