Governance of Services: A Natural Function for Agents

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

[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

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. Doing this well requires 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.

Keywords: Institutions; Multi-agent systems; Organizations; 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 organizations are actively engaged in converting their existing information systems or constructing new systems based on SOA principles. 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 characterized completely in advance. The use of *dynamically* discovered, configured, deployed, engaged, and maintained services has not been successful yet. The problem is that current service standards, which are necessary for widespread usage of services, are unable to describe anything other than the simple syntax and formatting of service invocations; they are thus insufficient for characterizing the rich usage and interactions required throughout the *lifetimes* of service-based applications, from discovery through maintenance.

In particular, service-oriented computing is intended to enable services to be discovered and enacted across enterprise boundaries. If an organization 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 are well beyond what was originally envisioned for service-oriented architectures. The features include quality-of-service (QoS) and contracts, i.e., service-level agreements (SLAs) among the participants. Moreover, to make this 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, if the introduction of agents increases the flexibility of service interactions, it also introduces a new set of vulnerabilities, due to uncertainty and complexity that characterize 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 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 service engineering.

Agents exacerbate the problems, while - surprisingly - also providing the only reasonable solutions to them. The autonomy of agent-based services makes them less predictable, but also enables them to self recover and to avoid deadlocks and livelocks, thereby making them more reliable. Their ability to learn can increase their robustness by being able to adapt to changing interaction environments, but also can increase their unpredictability. Their abilities to negotiate and reconcile semantics can enable them to re-establish connections and relationships among services 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 will elaborate on the motivation for higher levels of abstraction to specify dynamic SOA. In section 3 the components of our governance model are discussed. The potential benefits of our governance model are illustrated in section 4. The paper finishes with the first conclusions and discussion for future work in section 5.

2 Motivation

Our point of departure is the idea of a real-life service engagement [16, 2, 13]. 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 realized 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. For example, this 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 standardized ways. There is a presumption not only that the application 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 can understand that as a violation regardless of the low-level means through which they 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, recognizing 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 this reasons, and following common practice, it is convenient or even essential that we model the organizational 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 to 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 organization 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 organization. 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 Uniform Commercial Code and other rules of (commercial) encounter [15]. 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) and so on. Violations would expose each party to sanctions.

The governance of a service engagement is its administration, especially when carried out by the participants [16]. 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 characterize the requirements of the stakeholders precisely without having them over-constrained by operational details. In the above case, the initialization 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 realized 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 the violations of normative relationships within the Amazon scope, and the participants may suffer the consequences as a result.

Finally, service engagement should also be considered in the business-to-business case. Here we consider interactions between enterprises such as between an automobile manufacturer and a parts supplier. Each party is an organization 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 organization (that is, empowered with signatory authority) are necessary to instantiate the contract. However, the enactment of the contract requires the participation of sub-organizations and their representative agents lower down the respective corporate hierarchies. We can imagine this as a combination of delegation and assignments of the high-level normative relationship as well. 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-organization. The peers may be able to renegotiate details such as delivery times and trucks (or not – depending on their powers and authorizations within their respective organizations). If the engagement cannot recover within the organizations, 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. While this abstraction creates increased stability over time and flexibility of application for the norms, it also creates a problem when using norms as the abstract concepts in norms need to be related to the concrete events/concepts that occur on the technical service level the system. In [1] we have 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 Governance Model

Based on the issues raised in the previous sections, we propose a governance model for virtual organisations that comprises three levels (inspired by [3]) (1) organisations, (2) agents and (3) services. Organisations describe real-life engagements, their context, expectations and norms. The relationships between agents are defined by the organisations to which they belong, but agents are lead by their own reasoning abilities, desires and beliefs. Agents activate services in order to achieve their goals. Services, or compositions of services, are encapsulated in the agents that make them available to others.

This governance model distinguishes for each of the three levels (1) the knowledge (ontologies) and (2) the processes involved. In the ontology (per level) the concepts that

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

Table 1. Model overview

are used to define the three components *structure* (**S**), *function* (**F**) and *behaviour* (**B**) are described. Table 1 provides an overview of the model.

(Virtual) organisations have a *Representative*. The *structure* of an organization is described in terms of roles, values, etc. Organisations have their own norms and purpose: goals and ethical *function*. The Quality of Ethos determines the way organisations are perceived (their *behavior* is seen).

Agents have *Owners*. Agents have their own level of autonomy, communicate with other agents, and make individual or collective decisions. They have their own individual goals, mores and values, beliefs, desires and intentions. Quality of Character defines the way they are perceived, determines their reputation.

Services are provided by *Service Providers*. Agents activate services 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 will describe each of the levels and their role in the governance model in more detail.

3.1 Organization

One of the main reasons for creating organizations is to provide the means for coordination that enables the achievement of global goals. Organizational structure has essentially two objectives [5]. Firstly, it facilitates the flow of information within the organization in order to reduce the uncertainty of decision making. Secondly, the structure of the organization should integrate organizational behaviour across the parts of the organization so that it is coordinated.

The design of organizational structures and processes determines the allocation of resources and people to specified tasks or purposes, and the coordination of these re-

sources to achieve organizational goals. Both in human enterprises as in multi-agent systems the concept of structure is central to design and analysis of organizations [4].

Williamson argues that the transaction costs are determinant for the organizational model [20]. Transaction costs will increase when the unpredictability and uncertainty of events increases, and/or when transactions require very specific investments, and/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, non- repetitive and require no transaction-specific investments, then the market is the optimal choice. Powell introduces networks as another possible coordination model [11]. Networks stress the interdependence between different organizational 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 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/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, cooperation depends solely on perception of mutual benefit.

Each of these types of organizations comes with a set of predefined patterns and organisational roles. Thus, depending on the context in which the SOA is developed a certain organisation type can be chosen that fits best. For instance, the automotive example probably needs a network organization because partnerships range over sequences of transactions and trust on timely delivery of parts is important. The Amazon market place is a good example of a market organisation, with its ensuing norms and roles.

3.2 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 representative of an organization. This feature allows for a hierarchical specification of an organization in terms of divisions, departments, etc. The example in the next section will illustrate this point where the car factory is part of a larger car manufacturer.

In complex organisations coordination between agents is mandatory to manage dependencies in 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 the top-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, 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, these dependencies are crucial to their individual ability to perform. If information is to be shared, there must be a means with which this is achieved: e.g. shared memory, broadcasting, multicasting, message passing).

The agents main purpose is to achieve the different forms of coordination run-time. So, when unexpected events happen the agents can adapt to the new situation by rearranging and coordinating the changes. This is a big advantage over the service choreography where this should be done all at design time.

3.3 Service Enactment

Agents enact services to reach their goals. Services may be complex: agents may need to orchestrate their behaviour. They may need to adapt/inform an agent if something goes wrong. Which agent that is, will depend on the design of the system. 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 and to adapt appropriately.

Obviously, services can be clustered 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. A negotiation between the service requestor and providers could then ensue to establish an agreed upon QoS and formalize a contract. The requestors and providers would commit to honour the resultant contracts. These require agent abilities which are above those of regular services.

4 Illustrative Scenario

In this section a car manufacturing company is modelled using the structure outlined in Section 3. 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 cars. These aspects are described separately.

Stakeholders: Organisation	Knowledge (ontology)	Process
– Suppliers – Customers • Dealers • Individuals – Shareholders – Manufacturers – Sales	**S:** Board of Directors; Shareholders meeting. Negotiated agreements **F:** Produce cars suitable for market. **B:** Profit; Cost Control for customers; Safety; Timeliness, "Green".	Required to make a profit. Cooperation / Negotiate for resources and sales. Timely delivery of cars.

Table 2. Stakeholders: organization view

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

Table 3. Stakeholders: agent view

4.1 Stakeholders

Modelling the organisation of the stakeholders entails determining the actors, including customers, suppliers, manufacturers and sales. 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 shareholders meeting. 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 summarized 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. In 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 (summarized 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 car, the customer identifier, as well as preferences such as colour and interior. A service level agreement is used to define

Stakeholders: Services	Knowledge (ontology)	Process
Ordering Service (Seller)	**S:** order(Order ID, Car Type, Preferences, Due date, customer ID) **F:** Orders a car of a specified type from the manufacturer and species the desired due date. **B:** SLA defines the acceptable time allocated to manufacturing and delivering cars. This is different to the due date as it may specify penalties for non-compliance.	Enactment of the service.

Table 4. Stakeholders: ordering service view

Stakeholders: Services	Knowledge (ontology)	Process
Return Service (Seller)	**S:** return(Car Type, Order ID, Customer ID) **F:** Return a defective car back to the manufacturer. Defective cars can also include cancelled orders, changed orders etc. **B:** SLA defines how long a refund should take as well as the "restocking fee" that applies when order is cancelled for non-functional reasons.	Enactment

Table 5. Stakeholders: return service view

the acceptable quality of service guarantees that have been negotiated. These include absolute dates for manufacturing as well as containing penalties for non-compliance.

Another service that is used is a return service (depicted in table 5) that describes how cars are returned to the manufacturer if found to be defective. Defective cars may be physically defective or cars that are no longer required due to the customer changing their mind. SLAs define the service parameters for accepting a return, as well as the possible fees for returning cars that are no longer required yet are functionally correct.

4.2 Manufacturer

The manufacturer is one of the stakeholders, as well as operating agents in the previous model. Examining the structure of the manufacturer allows a more specific model to be

Manufacturer: Organisation	Knowledge (ontology)	Process
– Suppliers – Shareholders – Board of Directors – Workers	**S:** Building cars. **F:** Produce cars for sale. **B:** Safety; Efficiency; Suitability; Cost Control.	Required to efficiently produce cars using supplies to specifications. Sufficient parts should be supplied, without large stock.

Table 6. Manufacturer: organization view

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

Table 7. Manufacturer: agent view

created (see table 6). In this case, the organisation is made up of suppliers, the manufacturing workers, as well as the controlling entities — the board of directors and the shareholders. In this case, the structure is in place to actually manufacture cars for the sales aspect to sell. The requirements here are to safely and efficiently produce the cars while managing the costs associated with them.

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

One service that the manufacturer requires is a service to manage supplies of car 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.

As can be readily seen from the above examples, the model described in Section 3 provides the ability to capture the details of a business with many levels of abstraction.

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

Table 8. Manufacturer: assembly service view

5 Discussion & Conclusion

Technology and economics are together driving the development of open systems: one pushing and the other pulling, but as yet their realization remain 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 issues, highlighting the (often complementary) research that is taking place in different areas of computer science and that we believe can be brought together to close the gap between the current state and the effective delivery of open systems, through the adoption of a governance perspective on the remaining challenges.

CORBA, and its like, has offered the primary approach to systems integration — whereby we mean joining legacy systems with new ones and the on-going maintenance of current systems — for the last two decades. While this has provided a means to connect, but decouple, a variety of software components, the emphasis has typically been on delivery *within* an organization. During this period, web services have appeared and we see some migration to this technology, but more as a substitute, again within an organization, rather than cross-organization service provision in line with the original vision. Some of the factors, as discussed at greater length in earlier sections, that we believe discourage uptake include: operational rather than functional service descriptions, difficulty in monitoring service provision, problems in locating and handling faults and determining responsibility in the case of incomplete delivery of the service.

The scientific computing revolution has centred on the development and evolution of the grid, which has adopted web services despite the issues 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, characterized by long-running programs and complicated workflows of several such programs, have driven the development of distributed workflow engines (e.g. Taverna [10]), YAWL [18], WS-BPEL [9] 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 [14]. Service level agreements are playing an increasing role in grid computing, not just as a way for a service consumer 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 [12, 19] the best use of resources, 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. [7, 6] who examine business value networks and build on WS-Agreement [8] and Unger et al. [17] 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 realization 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 open systems, both of which enforce 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 organizations will 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 agents, as a 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 4, the development of formal approaches to organizational modelling. These last offer the opportunity to construct machine-processable policies that capture high level organizational intentions and, yet whose influence reaches down to individual decisions such as that highlighted in the preceding paragraph.

Looking back at the above, we identify (i) dynamic behaviour, (ii) formalization of business roles and rules, (iii) response to change (over short and long term) and (iv) formalization of agreements (in the physical and the virtual world), as the key challenges to be met to achieve the next level of aspirations in electronic service provision. We believe it is clear from this necessarily partial view of a broad range of research, that

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

good foundations exist on which to build the next steps in delivering service-oriented architectures — as long as we are prepared to borrow, extend and collaborate, rather than re-invent.

In conclusion, what we have offered here is a broad assessment of the state of service oriented architectures, 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 organizational 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 service, workflow, 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 organizers and all the participants, and Christian Derksen in particular, for their contributions.

References

1. H. Aldewereld, S. Alvarez-Napagao, F. Dignum, and J. Vazquez-Salceda. Making norms concrete. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, 2010. To appear.
2. N. Desai, A. K. Chopra, and M. 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.
3. 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 155–164. ACM, 2009.
4. V. Dignum, F. Dignum, and L. 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.
5. R. Duncan. What is the right organizational structure: Decision tree analysis provides the answer. *Organizational Dynamics*, Winter:59–80, 1979.
6. I. Haq, A. Huqqani, and E. 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.
7. I. Haq, A. Paschke, E. 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.
8. D.G.A. Mobach, B.G. Overeinder, and F.M.T. Brazier. A ws-agreement based resource negotiation framework for mobile agents. *Scalable Computing Practice and Experience*, 7(1):23–36, 2006.
9. 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.
10. 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.
11. W. Powell. Neither market nor hierarchy: Network forms of organisation. *Research in Organisational Behaviour*, 12:295–336, 1990.
12. R Sakellariou and V. 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*. IOS Press, 2009.
13. M. P. Singh, A. K. Chopra, and N. Desai. Commitment-based service-oriented architecture. *IEEE Computer*, 42(11):72–79, 2009.
14. 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.
15. UCC. Uniform code council: The global language of business. Technical report, 2005.
16. Yathiraj B. Udupi and Munindar P. Singh. Governance of cross-organizational service agreements: A policy-based approach. In *2007 IEEE International Conference on Services Computing (SCC 2007)*, pages 36–43, 2007.
17. 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.
18. Wil M. P. van der Aalst and Arthur H. M. ter Hofstede. Yawl: yet another workflow language. *Inf. Syst.*, 30(4):245–275, 2005.
19. 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.
20. O. Williamson. *Markets and hierarchies: Analysis and Antitrust Implications*. Free Press, 1976. ISBN 13: 9780029353608.