A Business Service Selection Model For Automated Web Service Discovery Requirements

Tosca Lahiri[1], Mark Woodman[1]

[1] Middlesex University e-Centre
The School of Engineering and Information Sciences
The Burroughs, Hendon, London NW4 4BT

tl173@live.mdx.ac.uk; m.woodman@mdx.ac.uk

Abstract: Automated web service (WS) discovery, i.e. discovery without human intervention, is a goal of service-oriented computing. So far it is an elusive goal. The weaknesses of UDDI and other partial solutions have been extensively discussed, but little has been articulated concerning the totality of requirements for automated web service discovery. Our work has led to the conclusion that solving automated web service discovery will not be found through solely technical thinking. We argue that the business motivation for web services must be given prominence and so have looked to processes in business for the identification, assessment and selection of business services in order to assess comprehensively the requirements for web service discovery and selection. The paper uses a generic business service selection model as a guide to analyze a comprehensive set of requirements for facilities to support automated web service discovery. The paper presents an overview of recent work on aspects of WS discovery, proposes a business service selection model, considers a range of technical issues against the business model, articulates a full set of requirements, and concludes with comments on a system to support them.

Keywords: Web Services, Discovery, Interoperability, Requirements, Business Service Model

1 Introduction

The automated discovery and use of a web service (WS) i.e. programmed location, selection and use without human intervention, is a goal of service-oriented computing [1]. However, as discussed below, only partial solutions have so far been attempted.

Considerable work has been carried out to address the many aspects of automated web service discovery. However, no comprehensive, coherent set of requirements appears to have been published; only small subsets have been considered as solutions to automated WS

discovery. Clearly, a full set of requirements is needed if a viable solution is to emerge, one that takes account of interacting and conflicting requirements. We aim to articulate a full set of requirements from which it would be possible to build an appropriate mechanism (e.g. an infrastructure layer) that meets the requirements for automated WS discovery. It is particularly important to look at the whole requirements picture rather than just focusing on a particular problem facing web service discovery, because many requirements may be in conflict with each other.

This paper is organized as follows: we first review the scenarios for using web services and establish an appropriate perspective for automated discovery; next we examine the current technical solutions in WS discovery research, noting the requirements they implicitly or explicitly address; we then investigate a model for business service procurement as a guide for WS discovery – specifically as a guide to the type of requirements generally involved in WS discovery. This leads to focussing on seven areas for WS discovery. In the conclusion we consider a mechanism to support the requirements raised in this paper.

1.1 Automated Web Service Discovery

First we interpret some terminology to facilitate later description: we use the term 'application' to mean software that has been developed to use other software applications (usually) owned by another party. We use 'whole system' to be the executing combination of the application software with its selected web services and any software that sits in between (e.g. needed to find or adapt to a WS).

Web services proponents use them in three scenarios: the first involves the application developer identifying a WS prior to designing their application, and developing it with the identified WS in mind. This predetermination of the service completely restricts the whole system to using that service – no automated discovery is involved. In the second scenario an application developer chooses a set of several services from which one will be selected as needed – possibly depending on availability. The choices are made before the (client) application software has been fully developed, but the scenario involves at least a degree of dynamic service selection during execution. However, no automated discovery takes place as the whole

system executes. Both the first and second scenarios are now commonplace and are used in situations where system developers or integrators use web services as remote modules, which could be developed in-house, and are so under the control or influence of the system developer.

The third scenario is where the application is designed without any prior selection of a set of web services, such that the system discovers and selects from any that meets its requirements during execution, essentially at the point when the system needs it. This third scenario is what we believe quintessentially describes how web services should be used, but it is the most difficult to achieve. In fact it is rare – and it wholly depends on effective automated WS discovery.

1.2 Technical WS Discovery Solutions

The goal of automated web service discovery is that the client software finds and selects the required web services during execution to make up the whole system; there is no human intervention at that point. In principle the whole system may behave differently on each execution, as it discovers and uses different web services.

The current solutions for WS discovery are partial, often because they do not distinguish between the three scenarios above. They appear not to consider the business ideas that underpin web services, particularly to do with ownership and control: they are purely technical solutions. Such solutions are not a sufficient basis for automated WS discovery.

1.3 Using a Business Service Metaphor

Our stance in this paper is that the weaknesses of the solutions so far are due to a lack of consideration to all that is required of a web-based business service developed independently by others. Since this is a situation routinely faced in business, and since the ideas of web services and SoA are supposed to reflect business, we propose to seek solutions for fully automated WS discovery by using business practices for service discovery and selection as a model.

We next examine the problems with current solutions before proceeding to characterize how businesses locate and choose services

which are needed for their business function, but which they do not or cannot supply for themselves, sometimes for financial reasons.

2 Current WS Discovery Solutions

Due to space limitations in the following brief review, we focus on a selection of recent contributions. For a review of older contributions see [2].

2.1 UDDI Enhancements

Because it appears to dominate perceptions about WS discovery, despite its well-documented deficiencies [3] we start with work on UDDI. It is important here to mention the use of the UDDI and in particular the work of Ran [4]. We argue that the UDDI does not allow for quality (what Ran calls quality of service) metrics nor does it have the capability to include semantic information. Ran proposes extending the UDDI. This is despite exposing the following weaknesses:

1. The UDDI is largely unregulated
2. The links in the UDDI are unstable
3. The entries are only functional, with the non-functional (i.e. performance) attributes omitted.

These three points, with the earlier two, are enough to suggest that, if we are to match business processes successfully, the UDDI should be replaced and not extended. It is essential that we have some mechanism that enables dynamic discovery and selection. Our proposed business service selection model aims to support these five requirements that have been found lacking in the UDDI.

Despite these failings, the UDDI is still being used. Aiming for efficiency, Lee *et al*. built a mechanism for service retrieval that was "implemented on a relational DB and cooperated with a UDDI registry" [5]. Their solution serves to highlight the problems inherent in the UDDI and as such will not succeed without taking into account other factors/problems such as the semantics and the WSDL descriptions that we discuss later.

Alrifai has proposed an "architecture" called iConnect that uses WSs to distribute data and how it supports "instant availability" [6]. This solution is also limited in that it too uses the UDDI. In particular it does

not use any semantic data, and it relies on the links in the UDDI being accurate. iConnect also relies on the WSDL file for a particular web service; the problem with this is that this file is deficient because it does not describe all the information needed for successful discovery. For example, the standard WSDL file lacks semantic descriptions. "iConnect" also needs human intervention to find a suitable web service.

Chen & Abhari [7] propose a new "framework" for dynamic service selection. However, this framework relies solely on the UDDI and as such encounters the problems we have previously discussed. As well as the fact that this framework is not yet implemented by Chen & Abhari we infer that not all the problems with web service discovery are addressed with this framework.

2.2 Other Solutions

Moving away from the UDDI, there are other attempted solutions to automate web service discovery. Rouached & Godart propose a "run-time service discovery process for web service compositions" [8]. The main weakness of this solution is that it disregards any web service that fails to meet precisely the application's functional and non-functional needs of a service. However, this approach does nicely encapsulate the requirement that a discovered web service can be adapted for use by software.

Ma, *et al*. [9] concentrate on semantics, emphasizing the view that irrelevant web services can be eliminated from search results based on the data the service returns. We infer from this and other similar work, a requirement that there is a need for a much fuller description. Otherwise, web services are disregarded only because the available description was not accurate enough. Thus a situation can arise in which a WS actually does what is required but would not be used because its description was lacking.

2.3 Distributed or Centralized Discovery Solutions

In this paper we have already considered some of the work related to the topic of WS discovery. In this section we will look at the important subjects of centralization, the UDDI and semantic information.

One issue for such WS discovery requirements is to do with centralization: must some middleware be interposed between requesting software and a WS, or must a peer-to-peer relationship between the two be maintained at all costs? Banaei-Kashani developed a discovery "architecture" that was based on peer-to-peer nodes: "WSPDS ... is a decentralized discovery service with peer-to-peer architecture for the Web Services infrastructure" [10].

This decentralized approach leaves itself open to misuse and brings about reduced service support. With a centralized discovery mechanism (whether called an "architecture", "framework", or "infrastructure") there is an opportunity to build confidence in service discovery because there is a record of web services and how they have performed. "Good practice" requirements come as a result. A common problem for business is that business people need to see some proof that a service is of value. A centralized approach can bring with it the means to provide for a community memory of web services and a community influence on those who provide them. Some centralized pooling of knowledge addresses many of the problems raised in previous sections of this paper.

The lack of business confidence is a common barrier to adoption of web services: "Current testing techniques for web services are unable to assure the desired level of trustworthiness, which presents a barrier to WS applications in mission and business critical environments" [11]. Taking a decentralized approach does not remove this barrier as there is a lack of metadata about a particular web service. In a centralized approach, with each component corresponding to a business practice, we can have different perspectives of use. With each perspective we have the opportunity to fine tune the discovery process. A decentralized approach does not assign ownership. Therefore, there is no business user available to be responsible for the service provided. This does not match business functions where some entity takes responsibility for the services it provides. A centralized approach aims to ensure consumer confidence.

2.4 Semantic Concerns

Regarding semantic information, Rajasekaran presents "an approach, which allows software developers to incorporate semantic descriptions of Web services during code development" [12]. We suggest that this is

not the best point to add semantic information to a web service: since we are discussing ontologies it would be more accurate to include semantic information at a higher level. We further suggest that the semantic information should be added when the candidate web service subscribes to the directory. This will take the responsibility away from the programmer and give it to a higher-level business process. In addition, this would give the semantic information a different perspective than that provided by the programmer of web services. The programmer does not necessarily have access to the higher-level business process so does not see the relevant semantic information.

We have investigated these recent and other (older) WS discovery solutions and our primary, general conclusion is that they fail to look at all the requirements for WS discovery. Our secondary observation is that even in the specific areas that have been focused on in these solutions there are failings, which we have highlighted. (The adherence to the UDDI mechanism is a problem.) Finally, we note that these (and other) discovery solutions do not look at the problem from the viewpoint of WSs matching business practices.

3 A Business Model for Web Service Discovery

If web services represent business processes and services in software, we argue that reasoning about business practices is more likely to help software-based practices for web services. To start with, web services are created by one company and made available to others under certain conditions, normally for financial profit. Therefore potential or actual consumers of a service do not directly control the functions or performance characteristics of a service they might use. If, as we propose, the software protocols for (software) web services need to some extent match those for (human) business services. To begin to articulate requirements for mechanisms to support automated WS discovery, we articulate practices in general business service identification and selection.

3.1 Generic Service Discovery/Selection

The process of a business finding a service to outsource some of its work to is routine, but complex; the discovery and selection process

will involve, at a high level of abstraction, something like the following:

1. articulating as precisely as possible what is needed from the 'outsourced' service by the consuming business;
2. locating possible services and their descriptions of what is offered;
3. checking on the precise meaning of advertised services,;
4. obtaining knowledge about the quality of a service via referrals and endorsements via human networks of service users;
5. short-listing potential services for selection;
6. checking that the external service and the internal processes can be made to fit and negotiating adjustments when needed;
7. determining and/or negotiating prices and estimating overall costs of service use;
8. making final selection – possibly as a primary choice with one or two backups, each with likely differences in prices and process fits.

By comparison, for the software involved in a business to find, and then determine the suitability of the encapsulated software of another business is equally complex, and not at all routine (yet). Rather than devise piecemeal solutions to automated WS discovery, we propose to look for holistic solutions.

3.2 Software Service Discovery

We argue that the whole system – the requesting software (the client software), the web service, and any mediating software – needs to match (but not mimic) the discovery and selection processes that humans carry out in business. With this aim in mind, and after having undertaken a review of past and present WS solutions, the problems inherent in the area indicate that any solution needs to include a set of interacting mediating components. By highlighting the problems, we offer the research community an analysis of the field that potentially indicates several courses of action, i.e. several types of mechanism which can be implemented to support automated WS discovery.

As we shall discuss below, web services are not being developed for automated discovery. Applications are not being written to find and select, during execution, software that is offered as a service and that is

written and controlled by others with whom the owners of the requesting software do not parley with in order to change the service.

Our starting position about service discovery and selection in service-oriented systems follows these general principles:

- Discovery of a service is not just about finding it in some sort of directory: people try to assess quality aspects by using referrals, asking for reference sites, looking for testimonials, etc.
- How a service is advertised may not convey adequately what using it means; detail is important in determining if a service provides what is needed – at least within acceptable parameters. In software this is about semantics.
- A candidate service may do what a business needs but requires that the client business interacts with it in a way that the client does not normally operate: some special procedure that adapts a client's operation to that which the service needs may be required. In software this is about syntax and adapters.
- If a client business needs to pass some of its assets (or its customers' assets) to the service, it needs to trust the service to look after those assets and not to compromise them in any way. In software this is about security.
- Over a period a client business may gather information on how well a service is meeting its needs so as to develop knowledge for the time when reassessment of service offerings is needed. In software this is represented by business intelligence.
- It may be in the interests of a client's business for the results of the first execution of a service to be available the next time the service is run, even if the service does not store such results.
- Many services are subject to guidelines, regulation or standards, although these can be used to obfuscate as well as elucidate; a client business may look for a reliable set of standards that properly constrain a service. In software we need to validate against standards for which compliance is claimed.
- During the course of finding and selecting a service, a business will look at whatever descriptions of the candidates it can find, whether or not the descriptions were furnished by the service provider; in many circumstances an expert will be consulted to see if he or she knows about apparently comparable services that have been rejected. In software this points to types of metadata and tools to analyze and support the metadata.

If web services correspond to business services and as such are to be the elements of a service-oriented system, software developers need a solution that enables their software to make software- and business-related decisions, and do the business- and software-based negotiations that a human might do now. This is an area that has been largely ignored because, we judge from the literature [13,14] that proponents of discoverable web services have not thought enough about the business analogy nor truly addressed the fact that a web service, just like a service in business, is outside the client's control. The lack of business focus has resulted in impoverished analysis of the requirements. We are looking at the requirements for web service discovery with the aim of dealing with both software and business decisions that need to happen. In this paper we articulate the requirements that will be needed to support web services discovery.

4 The Business of WS Discovery Requirements

Adopting a business-oriented stance, and with the business-related insights from technical solutions, we can now examine requirements that could form a complete set for supporting automated WS discovery.

4.1 General Requirements

Although the WS discovery requirements we are considering are expressed at a high level of abstraction, it is important here to consider some detail because of interactions among requirements resulting from wide distribution of problems involved in WS discovery. A WS is conceptually different from other encapsulations of business behaviour in software because it is outside the control of the requesting software, and, its behaviour cannot be changed by negotiation as it is generally when one business wants to utilize software originating in another. As Turner et al. [1] put it regarding SaaS, there is a separation of "the *possession* and *ownership* of software from its *use*".

There are technical and business reasons why we would want a system in which requesting software could find and use services without human intervention and across business boundaries. From the business viewpoint it would shorten the supply chains and enhance business–to–business cooperation. It is envisaged that software could

fully assess a potential service without human intervention – resulting in fewer delays while humans (inconsistently) assess candidate services in software. Automated WS discovery should result in more consistent business decisions; however, flexibility and adaptability are more highly valued in business over consistency, so those attributes must be realized by any WS discovery mechanism.

A major concern is for requesting application software to 'tell the world' that it needs a service. In general, this is not yet an automated task. The present situation is that the web service provider, in effect, says "this is what we do, do you want to invoke us?" This is done via the WSDL file. What is needed is a mechanism whereby the web service client says "this is what I need, who can service me?" Then we need to match the requesting software to the potential web services for subsequent use. According to a business model, what needs to happen next is that the list of candidate web services needs to be whittled down to those that are somehow most suitable. Part of the business-style suitability check is that standards requirements be considered: the WSDL files of the candidate web services would need to be checked to see if it complies with the W3C standard [15]. All these tasks are not automated; it would need the software developer to work manually through these steps.

4.2 Semantic Information

Web services are short on semantic information and as a result "pervasive networked devices and programs that can seamlessly interoperate are still a way off." [16] Nayak states that "Due to the lack of semantic descriptions of the Web services, the search results returned by the service registries are effectively inadequate." [17] This results in potential mismatches between requesting software and potential web service providers. A web service needs to bring with it a semantic description that can be interrogated by requesting software. These semantic descriptions are needed to develop domain knowledge about the web services and the requesting software. If two pieces of software have a well-defined semantic description it will be easier to make them interoperable. This will ensure that from a business standpoint the requesting software and the candidate WS are semantically matched. This does not happen at present.

4.3 Adapters

The next issue we have identified is in the use of syntactic adapters [18]. Again, without human intervention, it is possible that there will be instances when some requesting software will not interoperate with a web service because of a technical mismatch. From a business stance the syntactic adapters can cover instances where two parties with different forms need to interact. This is a common business problem where two organizations want to do business together but there is a point of contact that is not readily compatible. What happens now is that the software developers of the client-side have to negotiate with the software developers of the server-side to ensure that two pieces of software interoperate. The mismatches between service interfaces and protocols are hard to pinpoint [19]. This is a must-have requirement for WS discovery.

In order for a web service to support different interfaces and protocols there is a need to provide multiple interfaces [20]. At present the developers of the requesting client software are having to design and implement the interfaces that they need to interoperate with a particular web service.

4.4 Standards Compliance

WS discovery is affected by the use of standards or interpretations of standards. It is, from a business perspective, unhelpful to discover or select a web service that does not comply with application-required standards, or claims incorrectly to comply with required standards, or complies with ineffective standards. The client needs some form of assurance. It is up to the WS provider to offer evidence of conformance, or it is possible that the provider could sub-contract this process to another party.

The standards area relating to web services and SoA is unsatisfactory. There are too many standards organizations [21] all competing for the same ground, with each organization having its own set of beliefs and viewpoints. What consequently happens is that when a choice needs to be made about which WS to use the decision is biased because a particular viewpoint is favoured when the choice is made about which standard to comply with. The business client considers the influences brought to bear on any provider of a service. The same must

happen when choosing which web service to use. This approach matches the selection process followed from a business perspective.

With proprietary interest influencing standards for automated WS discovery, interoperability diminishes. In the business service selection model we propose that proprietary interests are negated as far as possible so that the following considerations are addressed: financial impact of employing a proprietary standard; a limited pool of knowledge if a particular proprietary standard is used; limit the effect of in-house commercial strategies driving standards down one track and not the other; bypass disputes concerning IPR; and avoid infrastructure limitations caused by implementing a particular proprietary standard. These business factors will all assist in automating WS discovery.

Another issue with standards is that they are seen as not being constraining enough. What would happen if a doctor prescribed a medicine that was unlicensed in a particular country? Due to the unknown side effects the patient may suffer, a third party regulatory body would be notified, would investigate independently to see if the doctor behaved irresponsibly and possibly take action. In this example, the laws are not strict enough to prevent this situation happening again (it is beyond the scope of this paper to discuss other ethical and medical reasons why this situation may occur). The same must happen from a standard's perspective. For example, Section 7 of the SOAP specification does not prevent software users from ignoring levels of trust. For the client this could have considerable impact on their business practices. This example from SOAP is a circular argument in that technically a business process must be implemented. If, for instance, Business A trusted Business B with the names and addresses of customers and, Business C obtained access to this data because the level of trust was ignored, then the results of giving away sensitive commercial data could be severe. Therefore, using the proposed business service selection model as a guide, we recognise requirements to tighten loopholes in standards that are used. The application (software) will have the option of asking for a service that complies with "Standard X" as it is, or, whether they want a third party to intervene and police "X", thus guaranteeing its constraints. In our example, this would mean strengthening Section 7 of the SOAP specification; for a definition of differences between standards, specifications, etc. see [22].

From a business perspective, the policing of standards will need to be considered as a requirement for automated WS discovery if the client application is to 'have faith' in the offered WS. What happens now is that a developer needs to spend resources making decisions about which, if any, standards to require of a potential WS. Therefore, it would be helpful if a business had the equivalent of a best practice document that leads them to the most appropriate standards body for their needs. The requirement then for the business service selection model is that it incorporates a mechanism that assesses the standards and categorises them according to their inherent features.

4.5 Security

We have already indicated in the previous section of this paper that there are problems with web service security. There are instances where the web service sessions are deemed insecure [23]. Namli and Dogac state that "… the privacy and security issues are indispensable for Web service technology in order to make them acceptable in more sensitive business transactions"[24]. If we want an automated web service discovery mechanism that matches business practice then we must address these security issues.

The requirements then for the security section of the business service selection model is that web services match the commercial, social and personal traits in a way that enhances web service interoperability via a discovery discourse. Our model takes a snapshot of the security needs of a client and then matches that to a web service. This matchmaking ensures that the expectations of the client are fulfilled so that we have secure sessions and levels of security that are acceptable to the client and the particular context in which they want to use the web service. For example, in our model there would be different levels of security according to the transaction type. If there is a financial aspect to the transaction, the level of security will be higher than if there were none. Our model addresses business-related security issues to ensure that the client is satisfied with the level of trust and security.

4.6 Business Environment Change

The next area of concern deals with changes in the business

environment. In §3 of this paper we sketched the way businesses typically select a service. A business's application software that uses another's web service will usually have to change if the prevailing business environment changes.

Automated discovery cannot be used in conventional web service collaborations that inherently have to deal with changes in the business environment. Even if the application software can adapt to a changing environment, people would have to be involved in selecting different web services for changes in functionality requirements. This would match the service selection procedures found in general business practice. There is a requirement then in our model for a mechanism that allows for changes in the business environment.

If web services can match the processes involved in coming to business-related decisions, the degree of interoperation increases, both in technical and business terms.

4.7 The Issue of State

In the business environment there is a concern about who owns and manages business objects that are used in the provision of products and services. If, for example, a business needs a service that retrieves publications based on a title and author search "looses" the specific title then all the publications by that particular author are retrieved; the resultset is too large and is inaccurate. Who is responsible for ensuring that the title of the publication is present in the search term? Is it up to the business that is requesting the service or do the providers of the service have to take responsibility for ensuring they have all the required data? We believe that there needs to be a third-party that manages the data on behalf of the service user and service provider.

State management is an issue for discovery of web services in that flexible and efficient procedures needs to manage state on both sides of the web service environment [25]. By this we mean the memory that a requesting application needs before and after using a service. In automated WS discovery there is a requirement therefore for a mechanism that manages the memory state of data.

The issue of state is about the ownership of data and who takes responsibility for ensuring its validity. From a business point of view will enterprises be comfortable with letting unidentified users (requesting software) access this data? Numerous matters need to be

considered when answering this question. Some like standards and security we have investigated in this paper. Others like social and political factors are beyond the scope of this paper. Our investigations lead us to believe that the issue of state management is a requirement in our business service selection model.

4.8 Metadata

Before a business will use a service provided by another business it takes into consideration many factors. These factors contribute to a large proportion of the decision-making processes. By this we mean the reasons why a business will use a particular service provider. For example, the length of time it takes to provide a service, or the reputation ranking of a particular service. In the business sense it is therefore a requirement to have data about the data used in a service.

To improve automated WS discovery it is an analogous requirement to have a repository of the metadata about web services; not just data about its operations and interfaces but a record of how it has performed. With this record, web services could be highlighted as "high performers". In our business service selection model it is a requirement that we hold structured data about WSs. This data would describe the characteristics of a particular WS. We would use this data the next time a web service becomes a candidate for use. For example, we would measure how long it took a particular WS operation to complete on a given day at a given time. With this execution rate recorded in a structure imposed by the business service selection model the next time the WS is considered for use it is possible to gauge whether it is fast enough. This speed of execution could also have implications for cost. That is one measure in our business service selection model that drives interoperability.

5 Conclusion

Web services are, and were always meant to be, a software encapsulation of business processes and, as such, they offer certain benefits, like interoperation with other software and software-based service discovery. The paradigm both implicitly and explicitly supports the possibility of businesses that have developed processes in software

to offer them as third-party suppliers to other businesses for use by their software-based processes. Unfortunately, the full set of business benefits are yet to be realized because WS discovery is not fully automated in practice. What we have observed in WS discovery solutions to date are what might be termed technical solutions in which idealized simplifications of finding and using services have been used. There has been a real absence of representations of business practice in solutions that try to support the discovery and selection of web services from third parties.

The main aim of this paper was to articulate requirements for a solution to the problems surrounding web service discovery by matching the general practices of business in discovering and utilizing services. We went back to the original idea of web services – to have a mechanism for software applications to utilize functionality for business processes provided by others.

We identified the requirements for service discovery by undertaking a literature review (part of which was included here). In this review we focused on what was hindering service discovery. This review brought to the fore requirements concerning semantic information, syntactic adapters, standards, security, business intelligence, data management and support services. Requirements for supporting interoperability also emerged.

The results of our research are the requirements that we have explored throughout this paper. We argue that it is important to address these points as a whole, not in isolation. There does not appear to be a solution to automated WS discovery that addresses these requirements as a whole.

Regarding future work: we are currently implementing a software mechanism, named the Web Service Architecture, which supports the requirements above. This work so far is showing that the benefits derived from the set of requirements are possible given certain design choices.

Our discussion in this paper has articulated the need to consider the whole set of requirements; requirements that must be interpreted according to a business sense. It is not sufficient to look merely for partial, technical fixes.

References

1. Turner, M., Budgen, D., and Brereton, P., "Turning Software into a Service," *Computer*, vol. 0018-9162/03, no. October 2003, pp. 38-44, 2003.
2. Fustos, J., "Web Services--A Buzz Word with Potentials," *USDA Forest Service Proceedings*, vol. 2006 2006.
3. Al Masri, E. and Mahmoud, Q. H., "Investigating web services on the world wide web," *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pp. 795-804, 2008.
4. Ran, S., "A model for web services discovery with QoS," *ACM SIGecom Exchanges*, vol. 4, no. 1, pp. 1-10, 2003.
5. Lee, K. H., Lee, K. C., and Kim, K. O. An efficient approach for service retrieval. Proceedings of the 2nd international conference on Ubiquitous information management and communication, 459-464. 2008. ACM New York, NY, USA.
6. Alrifai, R., "An architecture that incorporates web services to support distributed data," *Journal of Computing Sciences in Colleges*, vol. 23, no. 4, pp. 241-246, 2008.
7. Chen, Y. and Abhari, A., "An agent-based framework for dynamic web service selection," *Proceedings of the 2008 Spring simulation multiconference*, 2008.
8. Rouache, M. and Godart, C. A run-time service discovery process for web services compositions. Proceedings of the 10th international conference on Electronic commerce. 2008. ACM New York, NY, USA.
9. Ma, J., Zhang, Y., and He, J. Efficiently finding web services using a clustering semantic approach. Proceedings of the 2008 international workshop on Context enabled source and service selection, integration and adaptation: organized with the 17th International World Wide Web Conference (WWW 2008). 2008. ACM New York, NY, USA.
10. Banaei-Kashani, F., Chen, C. C., and Shahabi, C., "WSPDS: Web Services Peer-to-Peer Discovery Service," *Intl.Symposium on Web Services and Applications*, 2004.
11. Tsai, W. T., Wei, X., Chen, Y., Xiao, B., Paul, R., and Huang, H., "Developing and assuring trustworthy Web services," *Autonomous Decentralized Systems, 2005.ISADS 2005.Proceedings*, pp. 43-50, 2005.
12. Rajasekaran, P., Miller, J., Verma, K., and Sheth, A., "Enhancing Web Services Description and Discovery to Facilitate Composition," *Semantic Web Services and Web Process Composition: First International Workshop, SWSWPC 2004, San Diego, CA, USA, July 6, 2004: Revised Selected Papers*, 2005.
13. Vitvar, T., Mocan, A., Kerrigan, M., Zaremba, M., Zaremba, M., Moran, M., Cimpian, E., Haselwanter, T., and Fensel, D., "Semantically-enabled service oriented architecture: concepts, technology and application," *Service Oriented Computing and Applications*, vol. 1, no. 2, pp. 129-154, 2007.
14. Cauvet, C. and Guzelian, G., "Business Process Modeling: A Service-Oriented Approach," *Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)-Volume 00*, 2008.
15. W3C. Web Services Description Language (WSDL) 1.1. W3C . 2001. http://www.w3.org/TR/wsdl. Date Accessed: 16-2-2008
16. McIlraith, S. and Martin, D. L., "Bringing Semantics to Web Services," *IEEE Intelligent Systems*, vol. 1094-7167/03, no. January/February 2003, pp. 90-93, 2003.
17. Nayak, R. and Lee, B. Web Service Discovery with additional Semantics and Clustering. Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence, 555-558. 2007. IEEE Computer Society Washington, DC, USA.
18. Lahiri, T. and Woodman, M., "Web Service Architectures Need Constraining Standards: An Agenda for Developing Systems without Client-Side Software

Adapters," *Proceedings of the IASTED International Conference on Software Engineering*, pp. 45-52, Feb.2006.
19. Nezhad, H. R. M., Benatallah, B., Martens, A., Curbera, F., and Casati, F. Semi-automated adaptation of service interactions. Proceedings of the 16th international conference on World Wide Web, 993-1002. 2007. ACM Press New York, NY, USA.
20. Benatallah, B., Casati, F., Grigori, D., Nezhad, H. R. M., LIMOS, I., Campus des Cezeaux, B. P., and Aubiere Cedex, F., "Developing Adapters for Web Services Integration," *Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005: Proceedings*, 2005.
21. Bell, G., "A Time and a Place for Standards," *Queue*, vol. 2, no. 6, pp. 66-74, 2004.
22. Lahiri, T. and Woodman, M., "Web Service Standards: Do We Need Them?," in Pautasso, C. and Bussler, C. (eds.) *Emerging Web Services Technology* Springer/Birkhauser, 2007.
23. Bhargavan, K., Fournet, C., Gordon, A. D., and Corin, R., "Secure Sessions for Web Services," *ACM Transactions on Information and System Security*, vol. 10, no. 2 Article 8, 2007.
24. Namli, T. and Dogac, A., "Using SAML and XACML for Web Service Security and Privacy," *Securing Web Services: Practical Usage of Standards and Specifications*, 2007.
25. Song, X., Jeong, N., Hutto, P. W., Ramachandran, U., and Rehg, J. M. State Management in Web Services. [0-7695-2118-5/04], 1-7. 2004. IEEE. Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS'04). 2004