Optimising Performance in Network-Based Information Systems: Virtual Organisations and Customised Views.

Nickolas J. G. Falkner Paul D. Coddington Andrew L. Wendelborn
School of Computer Science
University of Adelaide
Adelaide, South Australia 5005
{jnick,paulc,andrew}@cs.adelaide.edu.au

Abstract

Network-based information systems use well-defined standards to ensure interoperability and also have a tightly coupled relationship between their internal data representation and the external network representation. Virtual organisations (VOs), where members share a problem-solving purpose rather than a location-based or formal organisation, constitute an environment where user requirements may not be met by these standards. A virtual organisation has no formal body to manage change requests for these standards so the user requirements cannot be met. We show how the decoupling of the internal and external representations, through the use of ontologies, can enhance the operation of these systems by enabling flexibility and extensibility. We illustrate this by demonstrating a system that implements and enhances the Domain Name System, a global network-based information system. Migrating an existing system to a decoupled, knowledge-driven system is neither simple nor effortless but can provide significant benefits.

Keywords: distributed system, information system, ontology, domain name system

1 Introduction

Any distributed system must have two basic characteristics: the system is composed of distributed elements and these distributed elements must be capable of successful intercommunication. Without the ability to transfer information between elements, the system cannot function. In some situations a distributed system may be controlled by a single organisation or a single administrative authority but, quite frequently, distributed systems may also be put together using the resources of a wide number of organisations. Where a large, and potentially changing, number of organisations need to use a distributed system, intercommunication standards must be defined so that host A can use the system to see host B's data without having to specifically negotiate the communication requirements with host B.

Throughout this paper we use the term *system* to denote a network-based information system which consists of a group of servers and clients. The servers store an informational resource in an internal representation and make it available to the clients, in a network representation, through a series of well-formed and well-defined messages. Clients request information from the server and the server responds with either the information or an explanatory message if the information isn't available. This explanatory message may contain references to other sources of the information.

Global distributed information systems must provide standards that define what type of messages may be passed through the system and the format of these messages. The traditional name for this network representation is 'wire format' as the internal representation of the data may not bear any relationship to the serialised form that is sent across the network. For example, the Domain Name System (DNS) has an internal representation and a very different wire format, the DNS message format[10, 11]. Internal and wire formats are linked by established translations, or serialisation mechanisms, and are tightly coupled. A change in internal format will require an immediate change in the translation mechanism and may also cause a change in the wire format. One of the objectives of this paper is to show why decoupling these formats leads to improved format conversions as well as the capability to provide customised services to clients in large network-based systems.

Depending on the translation mechanism, a change in the internal format could lead to incorrectly formatted data being sent out in wire format. For example, consider an internal data structure that comprises three contiguous variables, each a byte in length. The translation mechanism extracts the data using pointer arithmetic based on the data structure's start position in memory and a byte offset. Sup-

Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA'06)
0-7695-2641-1/06 $20.00 © 2006 IEEE

pose that the internal data structure is then changed to three 2-byte contiguous variables. In a big-endian system the translation mechanism will now read the high-end bits of the first variable as one component, then the low-end bits and, finally, the high-end bits of the second variable. These will then be pushed out onto the wire as variables one, two and three - with incorrect information. This arises because the relationship between internal format and wire format is based on structural information and an implicit understanding of the role of every component in every data structure.

Our solution is to annotate the data so that the semantic information, and potential structural translations, can be determined automatically. The mark-up of the internal format structure can then be used by a semantically aware translation mechanism to make the correct decisions in translating to wire format. Ideally, changes in one format do not require any explicit changes in the other format providing that the semantic and structural description is correct. At this point, the wire format is decoupled from the internal format as no assumptions have to be made about internal structure to achieve the desired output.

Decoupling the formats allows us to develop multiple translation pathways which can provide a wire format that matches the system requirements rather than restricting it to a single format that meets a single standard. Multiple wire formats can co-exist on a single server without any implications for the internal data format and can have a very low programming overhead if the translation mechanism is self-describing and easily manipulated.

In this paper, we will show why flexible multiple-format systems are valuable in network-based systems, how they can be implemented and their benefits. We also briefly discuss our example system, a multiple-format version of the Domain Name System (DNS)[10, 11].

2 Motivation

New technologies, such as the semantic web and workflow systems, demonstrate a need for more customisable solutions to network-based information system implementations. We develop our argument by examining the semantic web and workflow systems and establishing areas of inflexibility and potential expansion.

The semantic web is an example of enhancing a distributed information system, the WWW, with metadata annotation to allow improved reuse and data sharing [4]. Metadata is, at its simplest, data about data and can be as straight-forward as the name of a file or modification time, or as complex as a detailed schema to show where this piece of data sits in a large global data hierarchy. Metadata relationships, classes and properties can be captured in taxonomies or, with more work, as ontologies. An ontology is an explicit representation of a conceptualisation of a knowledge domain, where the domain is represented as a taxonomy, or hierarchy, with associated logical relationships which can be used to deduce information. The semantic web allows the use of metadata to place a structure on the data stored and used within a system and to show the relationships between this data. Ontologies can then be used to interpret the metadata and classify it to provide a truly machine-interpretable form of the data.

Workflows are formed when existing resources are linked in a specified order to achieve a task[12]. Workflow systems define relationships between the components in order to solve problems such as, in bio-informatics, the identification of defined protein groups and significant fragments in DNA. This draws upon a large number of information resources, both locally defined and networked, to allow the researcher to use the information effectively and efficiently. These resources can be information repositories, such as gene or protein databases in bio-informatics. They may be transformational entities that assemble or decompose information streams, which may include meta-data. The key characteristic is that the output of one resource is sent to the input of another resource, until the final output is produced. Problems may occur when attempting to connect two resources that have incompatible data syntax or, a larger problem, incompatible data semantics. When the resources cannot be modified the transformation between one data form and another can be a difficult problem. Solutions exist which use a semantic alignment check to then determine if a structural transformation may be carried out[1] but this is a semi-automated procedure and explicit semantic alignment, using semantic types in the cited work, has to be carried out before any structural translation can occur.

Workflow systems and the semantic web have a similar approach to the addition of metadata to components that have no pre-defined metadata. The semantic web can be used to automatically annotate the data streams from these 'dumb' components through the use of wrapper sites and enhanced gateways that provide at least some of the required, and missing, information[8]. A similar approach is taken when composing workflows from existing components. The workflow designer wants to store provenance or identification information with a data stream emanating from a component with no semantic metadata. In the case of workflow systems, the distance between the components is very small and, because of the direct connection between one component and another, any translation must be specifically designed in the workflow - it cannot be added as an ad-hoc intermediary as it can be in the semantic web.

Although we are not producing a relational database (RDB), we can note some useful concepts in the RDB community. The RDB community's definition of *views* provide a way of looking at the different ways that a database can be presented to its users through the use of stored relationships

and the conjunctive association of a number of stored relationships [14]. We adapt this terminology to include the information presented to a given user within a distributed system. For example, consider the widely distributed network-based information service that is the Domain Name System (DNS). The DNS clients and servers exchange DNS message format messages to ask and answer questions regarding domains.The result of a query to the DNS, presented in DNS message format, would be seen as a user's view of the DNS system. They do not have access to the internal format, despite the tight translational coupling between internal and wire formats. The term *wrapped resource* or *wrapper*, denoting a resource that has a translational wrapping provided through some external mechanism, and *mediator*, for an aggregator of resources (wrapped or otherwise) are also from the database community[14] and are used here without modification.

A distributed system conforming to a globally defined standard implicitly provides a global view shared by all users. Also, a user cannot change the global standard without potential impact upon all other users. So the global view is usually protected from users and cannot be altered on an ad-hoc basis. Since only an end-user knows their own requirements, this lack of flexibility can have a significant impact upon a user's ability to use the system in a way that is sensible, efficient and effective from their point of view. Without a good, or legal, reason for restricting access to the form of the information offered in a distributed system there is no good reason to restrict all users to a single view.

A virtual organisation (VO) is a group of organisations or individuals who have come together to share and use resources in a way that solves their problems[5]. Although widely used within the Grid community, there are many implicit virtual organisations that already exist with current distributed systems. A VO may have requirements that are not met by the current distributed system but won't be able to make a change in a widely used system as the VO may not have either the member numbers or organisational status to request a change.

Workflow systems composed of legacy items can be considered as a collection of small databases, wrappers and mediators where the views of each database resource are essentially fixed and must be made available through mediators and wrappers to meet semantic and syntactic requirements. The semantic web is more flexible but does not necessarily provide enough reflective or mutative facilities to allow format adaptation to specific user requirements. All views, if there are more than one, are predefined. Placing sources without annotation into the framework of the semantic web still requires mediators and wrappers.

The additional resources required for mediation and wrapping will, ultimately, put a sizeable load onto any system composed of legacy items. The additional elements would have to be specifically recoded to deal with new formats and, in the worst case, systems would spring up that use duplicate data with a different format for communication. Duplicate data entry and storage, unless for redundancy purposes, is a poor way to provide multi-format access to the same data.

Some systems already provide an ad-hoc approach to providing alternative views of globally accessible data, such as the IP-address specific responses available from the BIND DNS server. These do not solve the underlying problem and, more importantly, do not provide an easy mechanism for sharing the views with other users. For example, even if I can see a particular view of the DNS from my IP address, how do I share that view with another user with a non-privileged IP address? How will we capture the knowledge inherently contained in the situation - that old user and new user are now considered to be equivalent for the purposes of access to the privileged view?

When the system is focussed on knowledge capture we can represent the system as a set of overlapping knowledge domains and still provide the correct functionality. The configuration and execution parameters are derived from the stored knowledge. Deriving the system knowledge by reverse-engineering the system behaviour is not guaranteed to capture the correct behaviour as any limitations, or flaws, in the design will give a false view of the system knowledge. More importantly, the wire format and the way that it can be used will most likely not represent the nature and possible uses of the internal data and a traditional server environment will not allow legitimate users the flexible access they may require to get the most out of the system.

Decoupling the wire formats will allow us to meet individual client requirements without compromising the service to any other client. If we also provide a reflective interface to the wire format definition then we can meet new requirements while the system is executing rather than using pre-defined formats. We can also alter internal representations and add data or data transformations to those systems we control without interfering with their interoperability or established client base. With the decoupling, we can provide user-specific views without compromising the global view when we have unlimited access to the internals of the component. We can also provide a more flexible and easily programmed mediator or wrapper in cases where we do not have access to the code of a legacy component.

3 Benefits

The immediate benefit of the decoupling, however it is implemented, is that the internal structure can be changed to suit internal requirements without necessarily considering the the impact on the wire format and external users. Programmers and designers can focus on the knowledge

that the system is supposed to contain rather than an immediately grounded design that must use data structures and well-defined translation mechanisms.

The benefits are greater when the decoupling is achieved using the annotation of semantically-rich metadata and ontologies. Any rearrangement that maintains the semantic content of the system will have a valid structural representation providing that the semantics of the representation mechanism support such rearrangement. Thus the internal format, or any mutually-entailed representation, can be used interchangeably and the internal structures can also be distributed through a multi-component system providing that they meet the entailment requirements[7].

The ability to provide any number of valid wire formats also removes the need to implement different front ends, or servers, for systems that provide the same services but with differing structural representations or semantics. For example, SOAP is a light-weight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework, providing a message construct that can be exchanged over a variety of underlying protocols[6]. SOAP is used extensively to implement and support Web services. A wire-format-decoupled service can provide a SOAP-wrapped message to requesting clients without requiring a separate wrapper or mediator. It can also provide unwrapped messages.

Additionally, more features can be added to a system without those features overriding those defined by standards or already in use by other users. These features can be added directly to the internal format without any effect upon the external representations. Only those users who need to know will be made aware that such a feature is available. New components can function as standard members of a system without their additional functions affecting their global behaviour: our implementation of the DNS demonstrates this.

Finally, we can potentially eliminate the need to run multi-protocol solutions over separate network ports if the wire formats are differentiable and the failure modes of a given protocol do not correspond to a valid message in another format. It is then unnecessary to punch multiple holes through a firewall or define new port mappings for services across the VO, as we only need to make one hole and can accept all of the protocols through that. The wire formats must be sufficiently different that the validity of a message and its associated structure can be determined in reasonable time and with high reliability.

4 Method

The process of turning an existing system into a decoupled, knowledge-driven system is not a simple nor an effortless procedure. The first stage is to describe the system in terms of the knowledge it contains, rather than as a set of data structures and transformations that may be used to infer the existence of system knowledge. The function and purpose of all components of the system must be related back to the overall function of the system and every data element must be placed correctly and precisely into the system hierarchy. This then provides the basis for the production of an ontology that describes the internal data formats, the data semantics and the external data formats - such as the wire formats or, alternatively, disk and database storage formats.

This knowledge base is then used by an underlying engine that can parse the stored form of the ontology and correctly match service descriptions to service instances using a system library and a simple, but flexible, command language to describe translation and transformation operations. There is no requirement to explicitly decouple the internal and external formats since their description in the ontology defines the different formats without any required relationship between them. It is only the transformational pathway that will ultimately link them. Any internal format can have any, or all, of the external formats associated with it.

The system knowledge in the ontology is used by the engine to determine both the contents of the knowledge base and the form of the knowledge base, internally and externally. Thus, even within the system, multiple views peacefully co-exist to provide data in the best format for all users.

Once the knowledge capture is completed, the system's operation must be checked against a standard system, or set of standards, with well-defined test cases. The complexity of most large distributed information systems precludes the possibility of testing all operations so the following testing pattern is suggested.

First, validate the ontology and ensure that is correctly formed. At this stage, test for the correct inference of membership based on logical relationships. Then check that requests can be made to, and answered by, the server in the correct format. Finally, check the behaviour of system components that are at critical points in the ontology and test for system behaviour in the face of malformed or malicious requests. Many large-scale systems already have well-defined test cases that can be used to demonstrate that a server is performing correctly and these should be used whenever possible. Getting the ontology right is an essential step and takes the most time once the parser and interpretation engine, that will support the ontology, have been produced.

Once basic operation is confirmed, additional features or views can be added to the system and validated separately. Standards-based systems must have a standards-based core and an extension framework so that the user-requested extensions do not affect core operation. If the standards-based core is compromised then the overall system integrity could be compromised.

5 Implementation

We have implemented our decoupled format system using the RDF and the Web Ontology Language (OWL)[13] in conjunction with an RDF parser and interpretation engine, written in Java[2]. Although it can be used for a wide range of network-based information systems, this example system implements a Domain Name System (DNS) server. For ease of validation, we have only implemented a subset of the standard DNS resource record types[3]. The internal representations are stored in RDF-XML and read into memory as triples using the JENA Java framework for the Semantic Web[9]. Once in memory, the interpretation engine provides external access to the internal representation in accordance with the external access section of the ontology and the explicit transformational links between internal and external representations.

We added additional DNS resource record types, not defined in the DNS standards, that could provide additional information to requesters. This demonstrated the capability of the system to provide additional, or different, services to users based on their membership of certain groups. To achieve this the clients had to be capable of parsing these new records and, hence, the clients were also capable of parsing and interpreting wire format into internal format and vice versa. Although not shown here, we conducted testing with enhanced and non-enhanced clients to ensure that the server acted as a standards-based server and could also serve the additional records. The server correctly provided standard and additional records to enhanced clients but provided only standard records to the unenhanced client. (An unenhanced client is incapable of requesting records that aren't defined in the standard and, even if it could request them, could not interpret them due to the way that the DNS resource record numbers are handled.)

We define two client types: a DNS client that has ontology parsing and interpreting and can access the defining ontology and a SOAP-based client that requests DNS messages inside SOAP wrappers to pass between Web Services. The SOAP format was added after the initial work had been done on the system and required minor additions to the format ontology and a link to the internal format. No server recoding was required. We include a third data set, which shows the performance of unenhanced client and server performing the same operations, for comparison. The third data set measures time spent to respond to DNS message format requests on the standard port, as it could not respond to SOAP format messages or messages on other ports.

In the first experiment, DNS message format is available on the standard DNS port (53) and an XML 1.0/SOAP 1.2 format is available on an otherwise unassigned low port (60). Clients can request data stored in the DNS server through either DNS message format or SOAP format. Since the requests are being served on separate ports there is no overhead in determining which message format is being used. We conducted testing of the performance with 1000 trials and the mean and standard deviation were calculated. The clients and the server are all Java-based and executing on the same machine so there are no variations due to network traffic. Figure 1 shows the collected performance measurements. All results are within 1 millisecond at the 95% confidence level.

Client type	Mean time to complete query (s)	Std Dev
Standard DNS	0.088	0.007
Enhanced DNS	0.085	0.017
SOAP	0.096	0.013

Figure 1. Multi-format service on separate ports.

Requests are serviced in very similar times since both requests require translation from wire format to internal query format and then, once a result has been obtained, translation of the answer back to wire format. The slightly greater time for the SOAP requests arises from the requirement to embed a well-formed DNS message inside a SOAP wrapper.

In the second experiment, both wire protocols are made available on the same network port - DNS (53). The same requests are sent for all three clients over 1000 trials under the same conditions as the first experiment. Figure 2 shows the collected performance measurements.

Client type	Mean time to complete query (s)	Std Dev
Standard DNS	0.089	0.009
Enhanced DNS	0.092	0.013
SOAP	0.101	0.012

Figure 2. Multi-format service on a shared port.

A shared-port multi-format system must detect the correct format as the message arrives, rather than depending on the incoming port for format information. In this case, this distinction is simple as the SOAP message must start with "<env:Envelope"[6]. This is not the start of a valid message in DNS message format. Once the bytes"<env:" have been read in, no further testing is necessary and the message can be handled as a SOAP message. If the"<env:" header is not seen then the message can be processed as if it were in DNS message format since the Envelope header is a mandatory part of all SOAP messages. The testing code

adds a small amount of overhead and these overhead increases apply to all queries since now everything must be tested prior to conversion to internal format.

The more complex the shared-port specification is, the longer it will take to service requests. Wire protocols that are similar will make this procedure harder and, in some cases, distinction between alternative protocols will not be possible and separate ports may have to be used. For example, format TestA is defined that is four data values represented as four bytes where the bytes can have any values. TestB is defined with two data values represented as two two-byte groups where the bytes can have any value. There is no test to determine whether a four byte message is of TestA or TestB format with reliability.

We obtain two key benefits by taking an ontological approach to implementing the DNS. The first is the ability to handle additional wire formats on the same server. The second is the ability to add user-requested ways of viewing the stored data without having any effect upon the global view. The new system is both more flexible and more extensible but is still standards-compliant.

Our prototype system provides performance of the same order-of-magnitude as the unenhanced system. This is an excellent result given that additional computation is being carried out for each query within the system. A more comprehensive test suite is under development.

6 Conclusions

Removing the explicit link between the internal format of data in an information system and the external representation increases the agility and flexibility of network-based information systems. Not only can the internal format be optimised for efficiency without necessarily affecting the external representation but external representations can also be added and manipulated without causing problems.

This decoupling also allows for increased customisation of systems so that a virtual organisation can use an existing system in its own way without their requirements having an effect upon all other users of the system. We have shown this in our implementation.

Finally, the knowledge representation of a system can be used to produce a working model of that system through the use of interpretation engines and sound knowledge-representation practices.

References

[1] S. Bowers and B. Ludäscher. An ontology-driven framework for data transformation in scientific workflows. In *Proc. of the 1st Intl. Workshop on Data Integration in the Life Sciences (DILS)*. Springer, 2004.

[2] N. J. G. Falkner. An Ontology for DNS Zone Files, 2005. http://www.dhpc.adelaide.edu.au/.

[3] N. J. G. Falkner, P. D. Coddington, and A. L. Wendelborn. Developing an Ontology for the Domain Name System. In *Proc. of the 4th Intl. Workshop on Web Semantics*, Copenhagen, 2005. IEEE.

[4] D. Fensel, W. Wahlster, H. Lieberman, and J. Hendler. Introduction. In D. Fensel, W. Wahlster, H. Lieberman, and J. Hendler, editors, *Spinning the Semantic Web*. MIT Press, 2003.

[5] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling Scalable Virtual Organisations. *International Journal of Supercomputer Applications*, 15(3), 2001.

[6] M. Gudgin, M. Hadley, N. Mendehlson, J.-J. Moreau, and H. F. Nielsen. SOAP Version 1.2 Part 1: Messaging Framework. W3C Recommendation http://www.w3.org/TR/2003/REC-soap12-part1-20030624/, June 2003.

[7] P. Hayes. RDF Semantics, 2004. http://www.w3.org/TR/rdf-mt/.

[8] D. Martin and et al. OWL-S: Semantic Markup for Web Services, 2004. http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/.

[9] B. McBride. Jena A Semantic Web Framework for Java. http://jena.sourceforge.net/.

[10] P. Mockapetris. Domain Names–Concepts and Facilities. Request for Comment RFC 1034, 1987. http://www.dns.net/dnsrd/rfc.

[11] P. Mockapetris. Domain Names–Implementation and Specification. Request for Comment RFC 1035, 1987. http://www.dns.net/dnsrd/rfc.

[12] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[13] M. K. Smith, C. Welty, and D. L. McGuinness. OWL Web Ontology Language Guide, 2004. http://www.w3.org/TR/2004/REC-owl-guide-20040210/.

[14] J. D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.