

Extending Conceptual Models for Web Based Applications ^{*}

Phillipa Oaks, Arthur H.M. ter Hofstede, David Edmond, and Murray Spork

Centre for Information Technology Innovation
Faculty of Information Technology
Queensland University of Technology
GPO Box 2434, Brisbane, QLD 4001, Australia
{p.oaks,a.terhofstede,d.edmond,m.spork}@qut.edu.au

Abstract. The next phase envisioned for the World Wide Web is automated ad-hoc interaction between intelligent agents, web services, databases and semantic web enabled applications. Although at present this appears to be a distant objective, there are practical steps that can be taken to advance the vision. We propose an extension to classical conceptual models to allow the definition of application components in terms of public standards and explicit semantics, thus building into web-based applications, the foundation for shared understanding and interoperability. The use of external definitions and the need to store outsourced type information internally, brings to light the issue of object identity in a global environment, where object instances may be identified by multiple externally controlled identification schemes. We illustrate how traditional conceptual models may be augmented to recognise and deal with multiple identities.

1 Introduction

The world wide web (WWW) has already made interaction between people and web pages possible. The next wave of interaction promised by the web is between software applications operating without the need for human intervention. These web-based applications could be federated databases, intelligent agents, grid services, Web services and other kinds of distributed applications.

This vision for web applications involves *loosely coupled* applications operating on different platforms and *interoperating without prior agreements* in place. There are several problems to be solved before the vision can become a reality.

The words, terminology and definitions used to describe a software application, by its designers and developers, are naturally biased towards their own context and naming conventions. This local terminology may not be understood by users outside of this environment. Shared understanding is a major problem

^{*} This work was supported by the Australian Research Council SPIRT Grant “Self-describing transactions operating in a large, open, heterogeneous, and distributed environment” involving QUT, UNSW and GBST Holdings Pty Ltd.

for ad-hoc interaction between web applications. One solution is to *reuse* common external definitions, standards and specifications to describe some aspects of an application. For example, the ISO 8601 standard can be used to describe and represent dates.

Unfortunately, universally accepted definitions such as ISO 8601 do not exist for all the things that can be described by web applications. There are efforts underway¹ to create specifications for “universal” descriptions of various aspects of web based business applications but it is unlikely that these one-size-fits-all specifications will be useful for all users in all contexts². In addition, there will be many web applications that are not business related.

It is more likely that vendors and user interest groups will generate many similar standards and specifications that suit their own needs. In this case, web application providers risk a problem similar to vendor lock-in. By selecting one particular standard over another, application providers reduce their user base to those that use the same standard.

A better approach would be to create local definitions similar to application profiles [1, 2] that refer to all the standards and specifications that seem to be appropriate in the application context. In this way, applications can be made available to a wide range of users and the reliance on any one particular standard is reduced. These local definitions, based on external definitions and standards, can evolve to incorporate new standards as they are developed and to reflect changes in existing standards and the application environment. At present there are no explicit methods for developing conceptual models for web-based applications that take into account external definitions and specifications.

In this paper we introduce *outsourced types*, an extension to conceptual models. Outsourced types ensure web applications are developed from the conceptual level with the capability to use and understand information specified elsewhere. An outsourced type is an abstraction mechanism that allows the conceptual modeler to delegate (or outsource) responsibility for the definition of the internal structure of the type. The modeler can also attach requirements for external services to the outsourced type, thus providing the ability to manipulate and query an instance of the outsourced type without having to understand its internal representation.

In the next section we present a conceptual model to describe a relatively complex case study to show how outsourced types can be used in a global financial services context. Sections 3.1 to 3.3 motivate the need for outsourced type descriptions with the main focus on how to provide unique identities for instances of outsourced types in a global context. Section 4 introduces a meta-model for outsourced types. In section 5 we suggest some simple questions that can be used to identify the kinds of objects that may be candidates for outsourcing and illustrate how conceptual models may be augmented to allow the generation of relational schemas for the storage of entities with multiple identities. Related work is discussed in section 7.

¹ www.oasis-open.org/committees/ubl/ and www.rosettanet.org

² lists.ebxml.org/archives/ebxml-dev/200106/msg00038.htm

2 Case Study

Object Role Modeling (ORM)[3] is a visual conceptual data modeling technique, mainly used for the design of relational databases. It is used to describe object types and their relationships in a particular application domain. ORM has an associated modeling methodology, the Conceptual Schema Design Procedure (CSDP). ORM and CSDP were selected to model the case study because of their ability to produce robust and graphic models that “rigorously capture the semantic nuances of an information system” (John Zachman in [3]).

We briefly describe some of the main concepts of ORM is to assist the reader to interpret the schema presented in figure 1. The ellipses represent entity types (e.g. *Request*), while the boxes represent roles. A fact type consists of one or more roles and can play a role itself in other fact types (e.g. the fact type *Collateral*). Double arrows represent uniqueness constraints (e.g. a *Request* plays or takes at most one role), while solid dots represent mandatory role constraints (e.g. every *Request* plays or takes a *Role*). When a string is put in parenthesis below the name of an entity type (e.g. *(id)*) this indicates the presence of a value type with a name which is the concatenation of that entity type name and that string. In this case instances of the value type uniquely identify instances of that entity type (e.g. *Requestid* is a value type providing identification for entity type *Request*). The schema in figure 1 also contains a sample population shown below the fact types.

The case study is drawn from the field of securities³ lending. In this domain, the owners of securities can lend those securities in exchange for cash collateral or other securities. Depending on the type of collateral, the owner will be recompensed by interest on the deposit of cash collateral, or a flat fee for non-cash collateral. Borrowers may have any one of several reasons to borrow securities, traditionally the most common, is to cover short sales⁴. Lenders usually participate in the transaction to generate additional income from their securities investments.

The model (figure 1) shows the information necessary to describe a request that borrowers or lenders could place in a market to indicate a willingness to borrow or lend securities. The request itself is not a web service but provides the *context* for several associated services such as *Placing* and *Retrieving* loan requests. Requests could be constructed in many ways depending on the capabilities of the applications making and taking the request. A stored request could be revealed as a web accessible form [4], serialized in XML [5, 6], RDF/RDFS⁵, or OWL⁶ documents, or it could be revealed gradually as part of an interactive information service.

The shaded ellipses in the figure represent outsourced types. Outsourced types are necessary for this application, designed to operate in a global environ-

³ Financial instruments or securities, in this context, are those traded on stock exchanges such as the NYSE, the LSE or the ASX.

⁴ Short sale - the sale of a security before it is purchased.

⁵ www.w3.org/RDF/

⁶ www.w3.org/2001/sw/WebOnt/

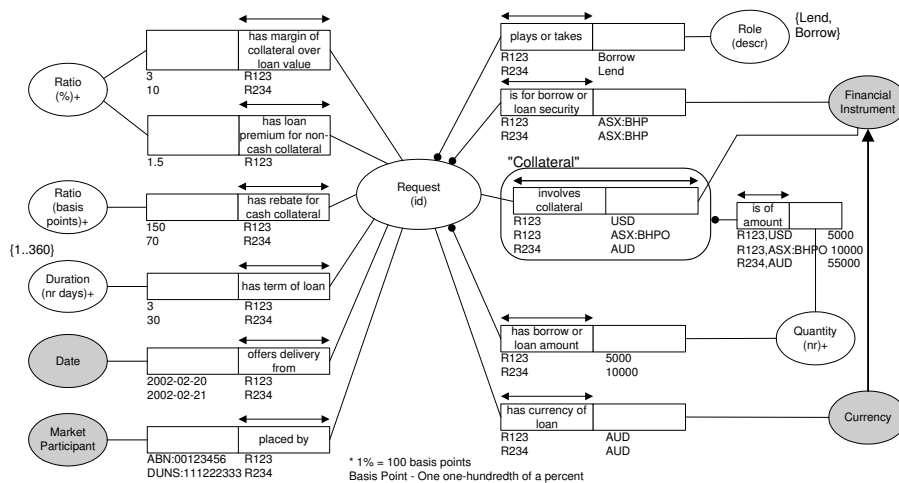


Fig. 1. Securities lending conceptual model

ment, for several reasons. The Date type (based on ISO 8601 [7]) is outsourced because date representations differ from country to country for example 10/7/03 represents October 7 in the USA and July 10 in Australia. The Currency type (based on the ISO 4217 standard for currency codes [8]) ensures there is no confusion when applications deal with money outside of their national borders.

Financial instruments are treated as outsourced types because they are subject to many different identification schemes and we want to ensure that all parties are talking about the same thing. This is especially relevant in the context of cross border securities trading where failed trades, due to mistaken identity, cost millions of dollars every year [9]. In addition, outsourced types allow service requirements to be attached to the financial instrument type for operations such as the retrieval of the name of its issuer, without having to have that information explicitly defined in the local model.

The population in the model (figure 1) captures two requests. The request (*R123*) is to *Borrow* the *borrow or loan amount* of *5000* of the *Financial instrument* identified as *ASX:BHP* shares⁷. The request offers two types of *Collateral*, the *Currency USD* (US dollars cash) and the *Financial instrument* identified as *ASX:BHPO* shares. The *margin of collateral over loan value* is how much more collateral, over and above the loan value, is required by the lender. The lenders return is the *loan premium for non-cash collateral* of *1.5%* on the *ASX:BHPO* shares and the *rebate for cash collateral* of *150* basis points⁸ on the cash value for a 3 day loan.

Over the duration of a securities loan, the value of the loan and the consequent value of the collateral required will fluctuate depending on the value of

⁷ ASX = Australian Stock Exchange, www.asx.com.au

⁸ Basis Point - One one-hundredth of a percent

the financial instrument and the current exchange rate between Australian and US dollars. The runtime re-calculation of the value of the loan means several operations are required to provide this information, a requirement for a service to retrieve the current market price of shares can be attached to the *Financial instrument*, and a requirement for an exchange rate conversion service can be attached to the *Currency* type.

3 Outsourced Types

3.1 Information Sources

A web-based application should be able to provide clarification of the terms it uses by referring to definitions, ontologies and other sources of information. The association of other sources of information with the outsourced type is primarily for the benefit of inference tools and mediators [10], it allows modelers to explicitly provide the semantics of the terms used in this context. As application terminology is often unique to a particular designer or software provider, support for dynamic runtime disambiguation and inferencing must be provided by the service itself. In addition, the use of common type systems such as external standards for the internal representation of data provides interoperability with all the other programs using the same standards. References to type definitions and alternative sources of information will enable inference engines to disambiguate the terms used in this context and is a practical step towards the goal of global interoperability.

The outsourced type *Date* demonstrates how external sources of information can be used. The outsourced type definition can refer directly to the ISO 8601 Date specification, or it can take advantage of the XML Schema Datatypes specification or schemas developed for use with RDF such as the Dublin Core (DC) element set⁹ both of which are based on ISO8601. For those applications that are not aware of the ISO standard or its syntax (ccyy-mm-dd), the outsourced type definition can also specify the requirements for services to translate dates to and from ISO format.

3.2 Service Descriptions and Capability Oriented Modeling

The lack of prior agreements between interaction partners means that there may be mismatches between the data required for a web service invocation and the data available. In this case service descriptions can be used to describe the operational capabilities that must be provided by or for the outsourced type in this context. Some of the operational capabilities will be generic, such as *creating instances*, *comparison*, *format translation* and *getting* and *setting* values, others will be specific to the type, such as retrieving the current market price from a financial instrument. A service description may also describe non-functional requirements and constraints. A description of these can be found in

⁹ dublincore.org/documents/1999/07/02/dces

[11]. At present, it is not possible to specify this kind of service requirement in ORM models [12]. Outsourced type definitions allow the specification of service requirements at a conceptual level, where the description is in terms of *what* is required (capabilities) rather than *who* by, or *how* the operations should be provided (implementation).

The delegation of tasks to external services is a natural extension of the object oriented and component programming paradigms. This kind of modularisation allows applications to concentrate on core competencies and delegate less relevant concerns to specialist services in non-core areas. At the conceptual level the modeler is only responsible for providing a description of what a service is required to do or provide. At runtime, the choice of which service implementation to use should be based on how well the service fulfils the required functional and non-functional constraints; rather than (as now) the functionality that can be inferred from a WSDL¹⁰ service description.

3.3 Identification Schemes

In the global environment there is often more than one externally controlled identification scheme and entities may have valid identities in several schemes. In the past, each application could provide its own unique identification mechanism for the local application context but it is difficult to maintain uniqueness constraints over relationships when the same entity may have several valid identifiers in different schemes.

Here we address the issue of entity or object identity rather than the issue of personal identity on the web covered by products such as the Microsoft Passport and frameworks such as [13, 14]. In UML models in particular, the identity of entities and objects is implicit rather than explicit as in ORM. The advantage of providing explicit identification is the ability to identify specific instances in terms that are meaningful outside of the system being modeled. An extension to the UML meta-model to allow explicit identity has been proposed in [15].

In ORM each instance of an entity must have an explicit unique identifier. An application that has outsourced entity types must be able to use different identification schemes *while ensuring that different identities for the same instance are recognised*, and the opposite case where the same identifier in different schemes, identifies different instances. The use of external identification schemes means the responsibility for enforcing constraints such as, a constraint that *identities within a scheme must be unique*, is delegated to the schemes' controller.

Each identification scheme will be owned or controlled by a different organization and each will have different rules and jurisdiction over the target group. The coverage of identification schemes may range from fully disjoint to fully overlapping depending on which aspects/properties/roles of the entity that the scheme is interested in. Similarly the scope of schemes may not be equivalent, for example the Australian ABN scheme [16] simply states that an Australian business exists and is registered (to pay tax), whereas a D-U-N-S¹¹ implies some

¹⁰ www.w3.org/2002/ws/desc/

¹¹ www.dnb.com

rating or assessment has taken place before the identifier is issued. Identification schemes themselves may have properties such as trustworthiness, accessibility, reliability and quality.

It may be necessary for some applications to limit the identification schemes that can be used; either by limiting the number of schemes, or by specifying acceptable schemes by name or by specifying the scope of acceptable schemes. The limitations on schemes could also be instance dependent, for example, if a Person is involved in a relationship “pays tax” and the person is from USA then only accept USA TFN’s or USA SSN’s to identify that Person.

In some cases it may not be possible to determine if an instance has an identity in a particular scheme, due to limitations on access to information without payment of a subscription or because of security concerns. Security is also an issue when it is necessary to pass identity information to downstream¹² services, there may be user constraints on whom the information can be passed to, or how the information can be used.

There are two conceptual modeling issues related to multiple identities that are a consequence of outsourcing types in a global context. The first issue is that there may not be an *exhaustive* list of all the possible identification schemes to start with, hence the model will not “know” about new schemes introduced after the modeling process is complete. Consequently a form of schema evolution is required to introduce new identity schemes. In terms of our meta-model (figure 2), an evolution mechanism for the introduction of new identification schemes would require information about the organization that controls, owns or publishes the scheme; whether or not its origin is based upon some other published information, such as a standard or law; and a source of information about the scheme itself.

The second modeling issue concerns ensuring the uniqueness *constraints* described in the base model are not violated when using instances of outsourced types. There are several options to prevent constraint violations at the implementation level. One option is to choose one or more of the external identification schemes to act as the canonical representation within the local application. A cover constraint is a requirement that all possible instances can be identified by the selected identification scheme or schemes. Therefore the schemes should be selected to maximise the coverage of instances and minimise the possibility of overlapping identities. The schemes should also have a similar interest in aspects of the instances they identify.

In terms of the case study, we could choose the ISO 15022 standard for International Security Identification Numbers (ISIN) [17] as it is an internationally agreed standard for the identification of securities. In practice we cannot use it on its own yet as it is not implemented in all countries. This means we would have to use ISIN along with all the identification schemes used in all the countries not yet covered by the ISIN scheme.

¹² Upstream services are users or clients, downstream services are providers of functionality.

Another option is to create an internal canonical identification scheme, and map identities from multiple external schemes to the canonical identifier, typically with the help of external conversion services. In most applications, including the case study, it will be necessary to record which identification scheme a client prefers to enable the reverse mapping back to the representation the client understands. A mechanism to do this is discussed more fully in section 6.

The advantage of this approach is that complete coverage can be guaranteed, at least internally. Complete coverage ensures the single canonical identification of any instance; and any instance previously unknown can be assigned a new identifier in the local scheme. The use of a single (locally) controlled canonical identifier also ensures that the constraints concerning the outsourced types are satisfied.

The identification of Financial Instruments is just one example of the multiplicity of identification schemes in the global context. In the first case above, several existing schemes could be chosen to provide complete coverage. However it will be difficult to find a manageably small number of schemes so that this can be done without overlapping identities. For this reason we tend to prefer the second option of a single canonical representation. The single canonical representation can be managed by a single party that has the ability to: ensure uniqueness constraints are preserved, include new identification schemes, and generate new identifiers when appropriate.

4 Meta Model

Figure 2 is a populated ORM meta-schema for outsourced types. Outsourced object types are a subtype of ORM Unnested Entity Types as defined in the ORM meta-schema [18]. They participate in three relationships which match the three types of information identified in section 3.

The first is the mandatory role *has instances governed by identification scheme*. The mandatory constraint means that there must be a way of identifying the scheme that issued the outsourced types' instance identifier. This rule makes explicit the entity identification rules inherited from ORM while recognising that instance identifiers may come from different identification schemes.

The second role is *has additional information*, this allows other information to be associated with the type, such as a definition in a thesaurus or ontology. The information is intended to be used to aid the comprehension of the syntax and semantics of the type and its roles in this context.

The central element in the meta-schema is *Information source*. The value constraint attached to *is of* shows an Information source can be of many different types. One type in particular, the *Service description*, is shown as a subtype of Information source because we want to indicate that it contains a particular kind of information and provides a link to the third role played by outsourced types, *has associated service description*. This relationship allows various requirements for downstream services to be associated with an outsourced type and its instances.

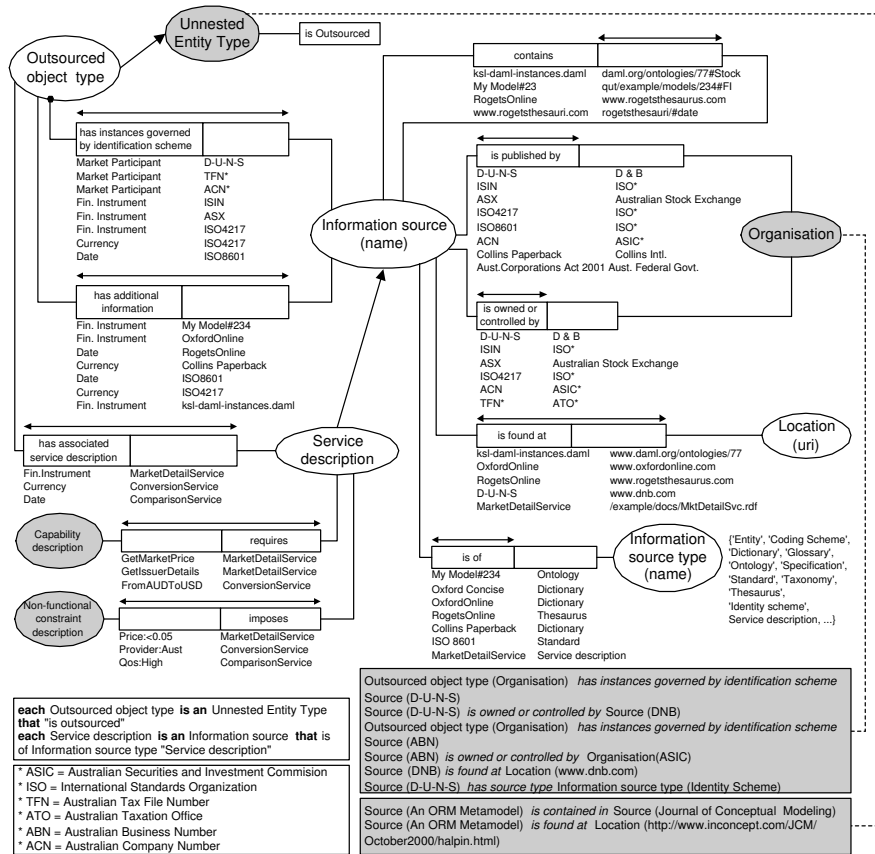


Fig. 2. Meta-model of outsourced object types

Information sources may also have other properties: they can contain or be contained within another source, they can be found at a particular location and information sources are published and/or controlled by some organisation.

The meta model describes the abstract syntax of the proposed extension. An abstract syntax captures concepts and their interrelations but does not deal with aspects of concrete representation. The shaded boxes provide a sample of a concrete representation. A concrete representation appropriate for web application descriptions is necessary but outside the scope of this paper.

5 Identifying Outsourced Types

Although ORM already has the *external type* construct to represent information found in another ORM model, outsourced types allow ORM models to include

external descriptions from many other sources in a similar manner to RDF or OWL ontologies and topic maps¹³.

The Conceptual Schema Design Procedure (CSDP) is a seven step procedure to transform facts related to the domain of interest into a well defined conceptual schema. Steps 1 to 3 are concerned with identifying entity types, value types and fact types. Steps 4 to 7 of the CSDP are when the constraints on entity types and the roles they play in relationships, are identified and incorporated into the schema diagram. The decision to use outsourced types can be taken when all the elements in the domain model have been identified, therefore the decision comes after step 3 of the CSDP.

Three questions help to decide whether an entity type should be defined as an outsourced type. At the conceptual level the concerns are to manage complexity, to reuse external definitions when they are appropriate and to prepare for interaction as a web-based application.

Elementary types: The first question is trivial and asks whether the object type represents an elementary or complex type. Elementary objects representing a number such as *Quantity* in figure 1 will not be treated as outsourced types. However, to promote interoperability elementary types could be defined in terms of an external definition such as XML Schema Part 2: Datatypes¹⁴.

Imported definitions: The second question asks whether there already exists a definition for this object type. These definitions can come from any model, schema or definition that is not the current model.

At the beginning of our investigation, we assumed all imported definitions should be treated as external, that is, supplied with an outsourced type definition for onward exposure to clients of the application being modeled. However as we have developed the meta-model, this requirement has become blurred. On one hand an imported element could be treated as an outsourced type only if it is also exposed to users. On the other hand, outsourced types can be used as a guide for software developers in the way an element should be implemented because the more external information is reused in local models, the greater will be the basis for shared understanding and interoperability. Reuse of publicly accessible information provides cost effectiveness, consistency and interoperability both within an organisational context and in the global world of web applications.

Exported definitions: The final question asks if the object type is to be exported or exposed to clients of this model. Exposure can either be direct or indirect. Direct exposure happens when objects are shared as part of the normal operation of an application, such as function names, parameters, return values or message components. Indirect exposure happens when up or downstream services require clarification of the semantics of the terms used by the application. For example a user of the securities lending request, may ask “what is a financial instrument?”, and the application could return the URI of a dictionary definition, or an ontology entry, or a list of alternative terms.

¹³ www.topicmaps.org/xtm

¹⁴ www.w3.org/XML/Schema

Outsourced types provide a means to describe entities in terms of external information, however web-based applications will need to store this information locally. In the next section we bridge the gap between the extension to the conceptual model and databases to store information about outsourced types. We illustrate how traditional ORM models must be modified to recognise multiple identity schemes.

6 Realisation

The existence of multiple external identity schemes is not evident from the representation of outsourced types in the local ORM model. This means the relational mapping procedure (Rmap) [3], used to convert an ORM model to a database schema, will be unaware of the multiple identities that may be associated with an instance of an outsourced type. The following examples illustrate how the model must be augmented as a precursor to Rmap. The augmentation procedure substitutes a canonical identifier for each external identifier and its associated identification scheme, while retaining information about which scheme is being used in each role. There are many cases that could be considered, each representing a different relationship (and its constraints) that an outsourced type can participate in with other outsourced and non-outsourced types. In this section we illustrate two sample cases that represent different uniqueness constraints on *binary* relationships between an outsourced and a non-outsourced entity type.

The augmentation process is done in two stages. In the first stage, a canonical identifier is introduced in place of an external identifier and its associated identification scheme. To retain knowledge about the relationship between: the canonical identifier, the external identifier, and its associated identification scheme, a ternary relationship (henceforth called the identity catalogue) is created. The identity catalogue is attached to the outsourced type, which now uses the canonical identifier as its primary reference scheme.

The identity catalogue is used to record *all* the identifiers and schemes that are known or can be found for a particular canonical identifier. There are two ways to build this list of schemes and their identifiers. The first option is to actively search for alternative identities and schemes when the database table is first created; the second option is to check and add new schemes when they are introduced to the application. The options represent a trade-off between the possibly lengthy time taken to populate the table when it is created and quick updates; or a longer time taken to make updates while new information is checked and incorporated into the table.

In the second stage, a new binary relation is introduced to make explicit the association of the canonical identifier and the identification scheme used in a specific role. The association of this new relation with the entities in the model depends upon the uniqueness constraints governing the whole relationship. This is illustrated below in relation to the one to many and many to many relationships.

Configuring a One to Many relationship. Figure 3(a) is in two parts, the top section shows the original relationships between an outsourced type (*Person*) and local entity types (*Event* and *Country*). A problem with this model is the identification scheme for *Person* is a combination of the name of an identification scheme and a value in that scheme. In the example, a *Person* may only represent one country but we are unable to determine from the identifiers, TFN:z024 and SSN:z021, that these really are different people. For this reason we introduce a canonical identifier (*cid*) for each outsourced entity instance and substitute the *cid* for the composite identification in the original *participates in* and *represents* roles.

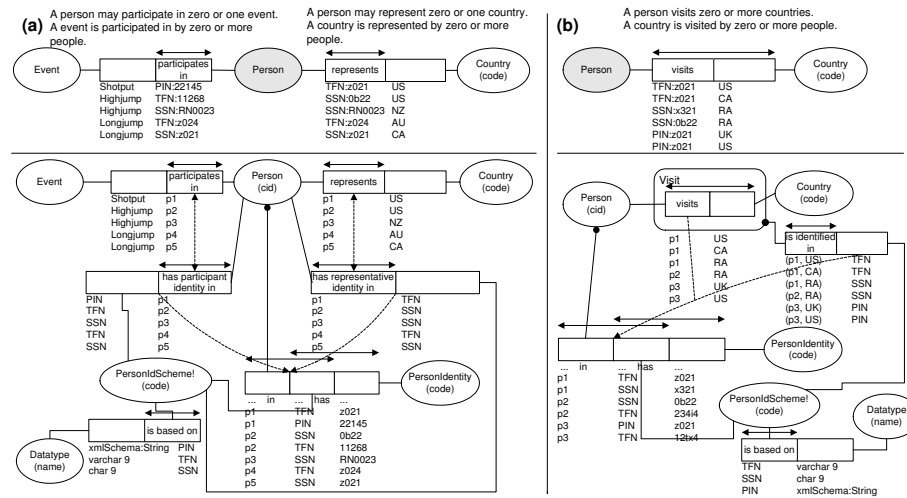


Fig. 3. One to Many and Many to Many relationships

The lower part of the figure is extended with an identity catalogue, to make the relationships between the *cid*, and all its related identification schemes and identities explicit. The identity catalogue is subject to two uniqueness constraints. The first is over the external identifier and its identification scheme. This constraint states that each identifier is unique within its identification scheme. The second constraint states that each combination of *cid* and identification scheme is unique. The combination of these two constraints means that a person with identities in different identification schemes can be uniquely identified by a single *cid*.

The next stage in the translation procedure is to make explicit the relationship between the *cid* and the identification scheme used in a particular role. The equality constraint between *participates in* and *has participant identity in* ensures that when a *cid* appears in the *participates in* relation, then this *cid* and the identification scheme used for this role are recorded in the new relation

has participant identity in. This relation records which specific scheme, amongst potentially many schemes, is relevant for the *participates in* relation. Similarly a new relation, *has representative identity in*, is constructed to represent the relevant identities and schemes for the *represents* relation. In both cases, as *Person* is on the “one side” of the one to many relation we attach the new relationships to *Person*.

Configuring a Many to Many relationship. Figure 3(b) shows a visitation relationship between People and Countries. The primary concern in this case is to ensure that each row of the visits/visited by relationship is unique. In the original schema we would be unable to enforce this constraint given the current information. For example, *TFN:z021* and *SSN:x321* are different identity codes, but are alternative identifiers for the same Person instance (p1).

Although there are alternative ways to prepare the many to many conversion for Rmap, we have elected to introduce an objectified relation *Visit* to represent the necessary information. The uniqueness constraint over the *Visit* roles ensures the original uniqueness requirement (that there are no duplicate rows) is maintained. A join subset constraint is placed over the roles involving a *Person* and a *PersonIdScheme* with the corresponding roles in the identity catalogue. This constraint means that for every fact in which a certain Person is used in conjunction with a Person Id Scheme, we should have information about this combination in the identity catalogue.

7 Related Work

There are two main strands of interest and activity in the Web services area. The first strand is the focus of several industry organizations including the Web services Interoperability Organization¹⁵ and the World Wide Web Consortium (W3C) Web services Activity¹⁶ which includes the Web services Architecture and Web services Description working groups. This strand is largely based upon the SOAP, WSDL and UDDI specifications. These services will, in the main, require off-line agreement between the parties over the semantics of WSDL interfaces and the message flow pattern before they can be used for automated interaction.

Web Services Description Language (WSDL) defines a grammar for describing Web service operations in terms of input and output messages. It has become the de-facto standard for the description of Web services and is currently (February 2003) in the process of being updated and revised by the World Wide Web Consortiums’ Web services Description Working Group¹⁷. WSDL does not provide facilities for referring to external information to describe the semantics of terms used in a Web service description.

The UDDI specification¹⁸ provides a mechanism primarily to advertise Web service providers and the services they offer. UDDI does not contribute to the

¹⁵ www.ws-i.org

¹⁶ www.w3.org/2002/ws

¹⁷ www.w3.org/2002/ws/desc/

¹⁸ www.uddi.org/

description of a service but it does provide “pointers” to tModels which represent technical documents such as specifications, protocols and categorisation schemes. Outsourced type definitions could be registered as tModels in a UDDI registry.

The second strand are semantic web services [19, 20], these services are on the boundary between the Semantic Web and Web services where semantic content and some intelligence are used in conjunction with the specifications listed above to provide a greater range of automated invocation and usability. In these services the interaction sequence will be determined by the goals of the user rather than the internal processes of the service provider.

The Web Service Modeling Framework (WSMF) [10, 21] is one of the interesting proposals in the area of semantic web services. The WSMF is based on two principles, the de-coupling of the various components that make up a Web service, and the use of mediators to translate between heterogeneous data representations and interaction styles. The information provided by outsourced types provides an explicit declaration of the semantics intended by the service provider, thus reducing the cognitive workload on mediators.

In the more general area of data modeling of web applications, Atzeni et. al. [22, 23] draw data from HTML web pages, and selectively rearrange and amalgamate it for presentation as new HTML pages. In contrast, outsourced types selectively draw information from a wide variety of sources in order to be able to describe aspects of a web application to users. This outsourced type information could be packaged and presented to users according to the methods described in [24–26]. The focus of that work is on providing context sensitive data and navigation options to human users but could also be applied in the area of machine to machine interaction.

Ontologies have been the subject of much work recently [27, 28] and they can provide high level views of a domain or lower level context specific views. One advantage of our method of extending conceptual models is that the ORM model can be used as the basis of a local ontology [29] or an application profile [2] that ties the local context to the global context via outsourced types. ORM provides the assurance of a methodology and modeling language that have been proven with time and experience to assist in building well formed conceptual models. The building of valid and useable ontologies will be better accomplished by using this accepted and proven conceptual modeling methodology.

8 Conclusion

In this paper we have introduced outsourced types, an extension to classical conceptual models. Outsourced types are a practical step towards achieving the goal of automated ad-hoc interaction between web-based applications.

Outsourced types provide several benefits. Firstly, they allow designers and developers to *reuse* existing definitions from local and global sources. Outsourced type definitions can also be reused as they are defined at a conceptual level outside of the scope of programming language, application and enterprise boundaries.

Outsourced types help to reduce the reliance on particular representations by allowing the use of alternative definitions (standards, specifications and ontologies) thus helping to avoid problems such as vendor lock-in. Another advantage of alternative definitions is that there will be a greater probability of interaction partners understanding at least one of the offered definitions. Alternative definitions also provide redundancy in a possibly unreliable web environment.

The outsourced type definition also provides the basis for downstream service composition by allowing specific statements of what capabilities the outsourced type is expected to provide for the local model.

The definition and creation of outsourced types will place a greater work load on web service providers but we believe the benefits will ultimately outweigh any increased costs.

References

1. Baker, T., Dekkers, M., Heery, R., Patel, M., Slokhe, G.: What Terms Does Your Metadata Use? Application Profiles as Machine Understandable Narratives. *Journal of Digital Information* **2** (2001)
2. Dekkers, M.: Application Profiles, or how to Mix and Match Metadata Schemas (2001) Cultivate Interactive, issue 3. Available from: <http://www.cultivate-int.org/issue3/schemas/>, (20 August 2002).
3. Halpin, T.: Information Modeling and Relational Databases: from conceptual analysis to logical design. Morgan Kaufmann Publishers, San Diego, CA, USA (2001)
4. Dumas, M., Aldred, L., ter Hofstede, A.: From conceptual models to constrained web forms. In: Real-World Semantic Web Applications. IOS Press (2002) 50–68
5. Demey, J., Jarrar, M., Meersman, R.: A markup Language for ORM Business Rules (2002) International Workshop on Rule Markup Languages for Business Rules on the Semantic Web, Sardinia (Italy) in conjunction with the First International Semantic Web Conference (ISWC2002).
6. Bird, L., Goodchild, A., Halpin, T.A.: Object Role Modelling and XML-Schema. In: Proceedings of the 19th International Conference on Conceptual Modeling (ER), Salt Lake City, Utah, USA, Springer-Verlag (2000) 309–322
7. International Organization for Standardization: ISO 8601:2000 Data elements and interchange formats – Information interchange – Representation of dates and times (2000) Available at: <http://www.iso.org>, (25 February 2002).
8. International Organization for Standardization: ISO 4217:2001 Codes for the representation of currencies and funds. (2001) Available at: <http://www.iso.org>, (25 February 2002).
9. London Stock Exchange: Exchange to introduce new global numbering system (2002) Press release, 7 November 2002, available from: <http://www.londonstockexchange.com/newsroom/releases>, (28 November 2002).
10. Fensel, D., Bussler, C.: The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications* **1** (2002) 113–137
11. O’Sullivan, J., Edmond, D., ter Hofstede, A.: What’s in a service? Towards accurate description of non-functional service properties. *Distributed and Parallel Databases Journal - Special Issue on E-Services* **12** (2002) 117–133
12. Weber, R., Zhang, Y.: An analytical evaluation of NIAM’s grammar for conceptual schema diagrams. *Information Systems Journal* **6** (1996) 147–170

13. OneName Corporation: Requirements for a global identity management service a position paper (2001) Presented to W3C Workshop on Web Services, San Jose, CA USA.
14. Hodges, J.: Liberty Architecture Overview, v1.0 (2002) Available from: <http://www.projectliberty.org/specs/liberty-architecture-overview-v1.0.pdf>, (10 October 2002).
15. Cranefield, S., Purvis, M.: Generating Ontology-Specific Content Languages. Information Science Discussion Paper 2001/08, University of Otago, Otago, New Zealand (2001) ISSN 1172-6024.
16. Australian Securities and Investment Commission: What's in a name? business names, company names, domain names and trade marks (2001) <http://www.asic.gov.au/asic/asic.nsf>, (11 July 2002).
17. von Rochow, I.B., Yous, N.: ISO 6166: Securities - International securities identification numbering system (ISIN) (2001) Information available from: <http://www.anna-web.com/>, (4 September 2002).
18. Halpin, T.: An ORM Metamodel. *Journal of Conceptual Modeling* **16** (2000)
19. Bussler, C., Fensel, D., Payne, T., Sycara, K.: Tutorial (T3): Semantic Web Services (2002) More information available at: <http://www.daml.ri.cmu.edu/tutorial/iswc-t3.html>, (15 October 2002).
20. Ankolekar, A., Burstein, M., Hobbs, J.R., Lassila, O., Martin, D.L., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T., Sycara, K., Zeng, H.: DAML-S: Semantic Markup For Web Services. In: Proceedings of SWWS' 01 The First Semantic Web Working Symposium, Stanford University, CA, USA (2001) 411–430
21. Bussler, C., Fensel, D., Maedche, A.: A Conceptual Architecture for Semantic Web Enabled Web Services. *SIGMOD Record, Special Section on Semantic Web and Data Management* **31** (2002)
22. Atzeni, P., Mecca, G., Merialdo, P.: To weave the web. In Jarke, M., Carey, M.J., Dittrich, K.R., Lochovsky, F.H., Loucopoulos, P., Jeusfeld, M.A., eds.: *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases*, Athens, Greece, Morgan Kaufmann (1997) 206–215 ISBN 1-55860-470-7.
23. Atzeni, P., Mecca, G., Merialdo, P., Sindoni, G.: A Logical Approach to Metadata in Web Bases. In: XI ERCIM Database Working Group Workshop: Metadata in Web Databases, Sankt Augustin, Germany, ERCIM (1998)
24. Ceri, S., Fraternali, P., Matera, M.: Conceptual Modeling of Data-Intensive Web Applications. *IEEE Internet Computing* **6** (2002) 20–30
25. Feyer, T., Kao, O., Schewe, K.D., Thalheim, B.: Design of Data-Intensive Web-Based Information Services. In: Proc. 1st International Conference on Web Information Systems Engineering (WISE 2000), Hong Kong, IEEE CS Press (2000) 462–467
26. Schwabe, D., Esmeraldo, L., Rossi, G., Lyardet, F.: Engineering Web Applications for Reuse. *IEEE Multimedia* **8** (2001) 20–31
27. McGuinness, D.L.: Ontologies and Online Commerce. *IEEE Intelligent Systems* **16** (2001) 9–10
28. Das, A., Wu, W., McGuinness, D.L.: Industrial Strength Ontology Management. In Cruz, I., Decker, S., Euzenat, J., McGuinness, D., eds.: *The Emerging Semantic Web. Volume 75 of Frontiers in Artificial Intelligence and Applications*. IOS Press (2002)
29. Meersman, R.: Ontologies and Databases: More than a Fleeting Resemblance (2002) in d'Atri A. and Missikoff M. (eds), *OES/SEO 2001 Rome Workshop*, Luiss Publications.