The Use of Patterns in Service Composition

Moe Thandar Tut and David Edmond

Cooperative Information Systems Group
Research Centre for IT Innovation
Queensland University of Technology
Brisbane, Australia
{*m.tut,d.edmond*}*@qut.edu.au*

Abstract. E-services are services that can be invoked over the Internet. One likely use of e-services would be to build business applications that can employ e-services from different service providers. This potential use of composite services in business settings highlights the issues of payment mechanisms, reliability, trust, inter-operability and service guarantees between different service providers. It also becomes essential to choose e-services that best fulfil the requirements of a particular business application. We investigate how patterns can be used in service composition to help in the development of business applications based on e-services.

Keywords: e-services, service composition, patterns

1 Introduction

In this paper, we investigate the possibility of using patterns to facilitate the composition process of electronic services. We will start with the discussion on e-services and service composition. Kotov (2001) describes e-services as "the realization of federated and dynamic e-business components in the Internet environment". One potential use of e-services would be to build business applications that can invoke e-services from different providers. Before a suitable composition model for electronic services can be determined, a clear description of an electronic service is needed. Electronic services offered over the Internet are also referred to as electronic services, web services, Internet services, web-based services or e-services. Piccinelli & Mokrushin (2001) define the concept of electronic services as "electronic visualisations of standard business services". A more technical definition is given as an interface that describes a collection of operations that are network accessible through standardized eXtensible Markup Language (XML) messaging (Colan 2001). One essential characteristic of e-services is the ability to be described, published, discovered and invoked dynamically in a distributed computing environment. The types of e-services can range from simple calculations and retrievals to complex business applications. The examples of electronic services include real-time stock quotes, content syndication, mapping services, payroll management, and credit scoring.

We would like to propose the use of patterns during the planning stage of service composition. Patterns represent a proven way of doing something. They could be business patterns such as how to model online store-fronts, or generic patterns such as project work patterns. However, the nature of patterns make the ideas generic or abstract. Hence, we would like to take the user through different levels of abstraction from generic to specific, that would result in the concrete model for business applications using e-services. We believe that this approach will result in a structured and a more intelligent search and composition, using both users and service providers' knowledge.

2 Service composition

To compose a comprehensive offer for their customers, successful companies focus on their core competencies, and outsource the support services to other companies. Piccinelli (1999) describes service composition as the ability to take existing services (or building blocks) and combine them to form new services. A composite service is one resulting from the integration, coordination and synchronization of different service components from two or more service providers. A composite service adds value that is not previously presented within the individual services. Piccinelli & Mokrushin (2001) suggest that the realisation of the full potential for the e-service vision depends upon on composition and interaction orchestration. By using electronic services to outsource services, the focus shifts from the connection to a specific business partner, to the definition of a specific business need. The types of business applications that can be developed using e-services from different providers include travel booking applications, portals and e-market palaces, on-line stores, supply chain management, inventory management, shipping and logistics, finance and insurance services. Service composition typically spans three phases: planning, definition and implementation. Yang & Papazoglou (2002) describe planning as the phase where the candidate services (elementary or composite) are sought and checked for composability and conformance. During this phase, alternative composition plans may be generated and proposed to the application developer. The outcome of this phase is the synthesis of a composite service out of desirable, or potentially available, matching, constituent services.

We would like to propose the use of patterns during the planning stage of service composition. Our objective is to facilitate the composition process and to support the developer in selecting components based on rich descriptions of services. Our assumption is that the business goal is to successfully compose a service, not to decompose the process model to the lowest level. We believe that the composer will prefer dealing with fewer providers and fewer e-services where possible. It would also increase the security and the trustworthiness of the resulting service. Naturally, when composing e-services, the functionalities provided by service components must be considered. We need to take into account the comparability of data types, message types, business sequence logics etc (Fensel

& Bussler 2002). However, our view is that service composition is much more than functional composition. Non-functional requirements also play a major part in the selection process for service components. Consideration should be given for non-functional requirements such as trust, reliability, security, geographical location, execution time and payment mechanisms. For instance, when composing a product-ordering e-service, we must also consider auxiliary services such as insurance financing, payment mechanisms, transporting and compliance with government regulations.

3 Patterns

Alexander (1979) describes a pattern as "a three-part rule, which expresses a relation between a certain context, a problem and a solution". Patterns help create a shared language for communicating insight and experience about the problems and their solutions in a particular context (Appleton 2000). A pattern catalog is a collection of related patterns (perhaps only loosely or informally related). It typically subdivides the patterns into at least a small number of broad categories and may include some amount of cross referencing between patterns (Buschmann et al. 1996). A set of connected patterns provides a framework upon which any design can be anchored (Salingaros 2000). Some examples on how patterns can be connected to each other include

- One pattern contains or generalises another smaller-scale pattern.
- Two patterns are complementary and one needs the other for completeness.
- Two patterns solve different problems that overlap and coexist on the same level.
- Two patterns solve the same problem in alternative, equally valid ways.
- Distinct patterns share a similar structure, thus implying a more abstract version.

The paper by Edmond & ter Hofstede (2000) discusses the use of libraries of common patterns of activity and instantiating and composing task structures from these patterns to enable service composition. The authors propose that well-defined frameworks within which services can be offered represent essential reliability mechanisms. Lord (2001) describes how to facilitate the application development process using the IBM Patterns for e-business. These patterns are classified as business patterns, integration patterns, composition patterns, custom design and application and runtime patterns (IBM 2002).

Patterns can be used to represent reusable business process logic for the applications. We would like to instantiate the generic patterns that are domain independent into specific patterns using the domain knowledge of the developer. Patterns will be stored in a repository and could be indexed based on a number of classification codes. The user could search the repository for appropriate patterns for the application. It might also result in modification / instantiation of patterns to fit the particular needs. Every task in the pattern should map to an e-service or

another pattern. The process will come to an end when the user finds the right level of abstraction for the process to compare with the service descriptions, taking into consideration the issue of trust, reliability and payment mechanisms. By doing so, the composition logic built into the pattern will be available to other users. It would also result in separation of process logic from implementation.

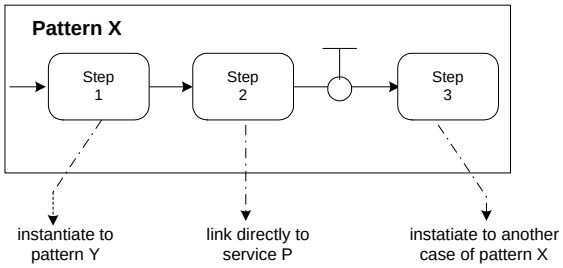

Fig. 1. Pattern instantiation

For example, according to figure 1, the services instantiated as part of *pattern X* must be configured within a particular three-step framework. According to this pattern, two steps must be invoked in sequence. After that, a decision will be made on whether to proceed to step 3, or to stop straight away. It can be seen that step 1 would be instantiated to another pattern called pattern Y. Step 2 would link directly to service P and step 3 would instantiate to another case of pattern X.

3.1 Generic patterns

In this subsection, we will introduce the two generic patterns, namely project pattern and maintain pattern, and illustrate how these patterns could be instantiated using domain knowledge with an educational service example.

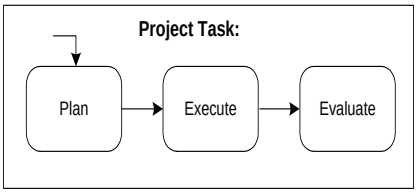

Fig. 2. Making and following a plan

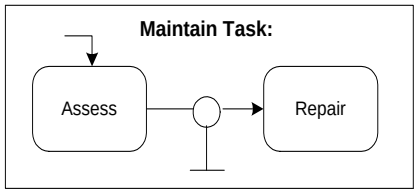

Fig. 3. Maintenance

The *project* pattern (figure 2) describes a systematic way of making and following a plan. This plan is used in such commonly arising situations as building

a house, developing software, and holiday planning. First we develop a plan of some kind, then we execute that plan, and finally we evaluate the outcome. Such a pattern is not instantiated in a thoughtless manner. Its use suggests a preparedness for deliberation, perhaps triggered by past experience, perhaps contrarily by the realisation that the situation is novel. This pattern highlights the preparedness for the task and evaluation of the task in addition to the execution. This pattern is generic in the sense that it could be applied to any task that would benefit from planning and review. The *maintain* pattern (figure 3), describes the process involved in assessment of a situation and making a decision to repair / improve the situation. This pattern could be applied in situations such as maintenance of some asset or improvement of processes. The use of such a pattern would seem to arise from the recognition of the possibility of damage. We can then employ this pattern as a means of recovery.

We will now consider how an educational service can be developed using these generic patterns. As described in figure 4, this service consists of a fairly generic, topic-free process of preparing a course of study, followed by the teaching and assessment of that course, and finishing with some kind of evaluation process. The entire process could be viewed as an example of the *project* framework discussed earlier: preparing the course is obviously a form of planning; teaching the course is a means of executing that plan; and, the evaluation of the course is clearly a form of evaluation.

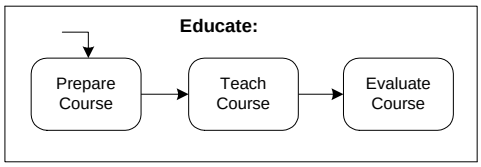

Fig. 4. Education as instantiation

The first stage of the process (prepare course) involves the decision-making from the user in terms of topics to include, resource availability, time constraints, etc. The outcome of this phase is the detailed plan with the particular topics and resources that will be used in defining the next process. The middle stage, the actual teaching and assessment, may be further refined into a process based on the outcome of the prepare course process. For instance, it could be instantiated as described in figure 5. In this case, it is represented as a process whereby three topics (SQL, ER modelling and workflow concepts) are taught, in sequence. Good teaching practice requires rapid feedback and the assessments are interspersed with the teaching, bringing the evaluation of student understanding closer to the actual teaching. For example, once SQL has been taught, two parallel streams are triggered – teaching ER and assessing SQL. A similar approach is taken after ER has been taught, with an additional requirement that the SQL assessment

be finished before the ER assessment begins. Workflow assessment is performed in the final exam, presumably. The final stage of the process (evaluate course can be viewed as an instantiation of *maintain* pattern as shown in figure 6. It involves the assessment of the course followed by a decision on whether or not to modify the course structure.

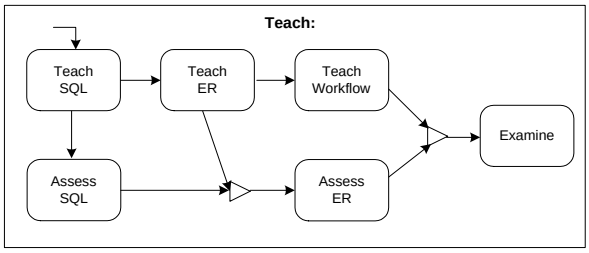

Fig. 5. Instantiation of teaching and assessment

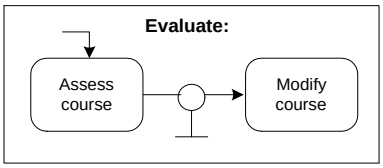

Fig. 6. Instantiation of maintain pattern

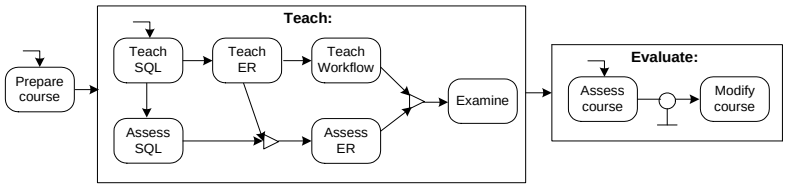

Fig. 7. Educational service pattern

The final process model for an educational service involving the development of a course with database concepts now looks like figure 7. Even though an educational service example is a simple one, it highlights the difficult nature of

decision making involved in constructing a service. Our instantiation of an education course is based on time and resource constraints as well as the application of business rules. The parallel processing model and synchronizers are instantiated with the domain knowledge about teaching a course. It can be seen that it is the result of using generic patterns to guide the decision of the composer by taking into consideration, the generic patterns, time and resource constraints, business rules as well as domain knowledge about education sector. The execution stage of this example can be represented as an e-service. It could also be a composite e-service that uses various on-line e-services to teach the individual topics. This process can also be considered as part of the process model for on-line educational applications.

Next we will consider the modification to the figure 7 when the teach process is outsourced. Imagine that we need to employ 3 lecturers to teach the topics in the course structure. In that case, we would need to associate each teach process with a payment process. Different payment mechanisms could be associated with teaching each topic. For instance, the lecturer who will teach SQL might like payment *after* his/her duties have been performed and the process could be modelled as shown in figure 8. The processes *TeachSQL* and *Payment to lecturer* are invoked sequentially. On the other hand, the lecturer who will teach ER would prefer payment in advance before teaching is carried out. Hence, the process will be modelled as in figure 9.

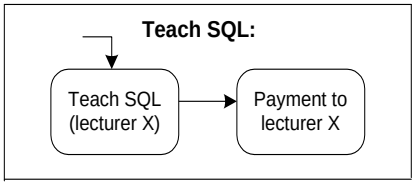

Fig. 8. Payment after execution

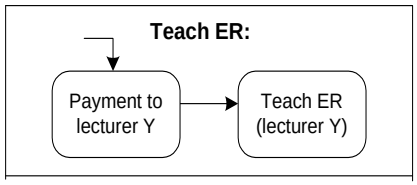

Fig. 9. Payment in advance

We believe that associating a payment with the execution of a service will be common in business applications using e-services from different providers. Hence, the different payment mechanisms employed by these providers will become one important criterion to consider when selecting service components.

3.2 Payment mechanism patterns

O'Sullivan et al. (2002) state that an essential ingredient of service representation is capturing the non-functional properties of services. These include the methods of charging and payment, the channels by which the service is provided, constraints on temporal and spatial availability, service quality, security, trust

and the rights attached to a service. Non-functional requirements are considered to play a crucial role in understanding the problem being addressed, the trade-offs discussed and the design solution proposed. Gross & Yu (2001) propose that non-functional requirements that are explicitly represented in design patterns aid in better understanding the rationales of design, and make patterns more amenable to structuring and analysis. We will now consider how non-functional requirements can be represented using patterns with the example of payment mechanisms patterns.

Most business interactions would result in a payment being made between service providers and service requestors. An organisation can employ a number of payment mechanisms based on the type of user, the type of industry and product, the type of accounting systems. We are interested in the processes involved in this payment pattern and when a particular process will be invoked. Payment mechanisms can be seen as interactions between three different processes: billing, payment and execution of service. The billing process represents the process of producing an invoice or a bill by the provider for the service. The payment process represents the process of accounting required when the payment is received by the provider. The service process represents the process of invoking the service requested by the user. Next, we will look at different mechanisms under which billing and payments could be carried out.

The example scenarios include subscription fees, cash sales, credit sales (accounts receivables).

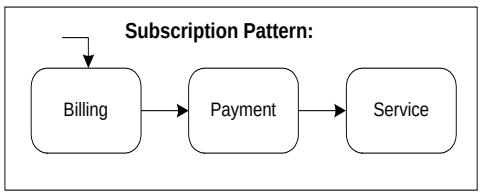

Fig. 10. Payment in advance: Subscription

- **Subscription:** This is a common payment scenario for membership subscriptions, magazine subscriptions and insurance premium payments. The user is expected to pay in advance before the service is performed. There is a certain order to the interaction as in figure 10. In this case, it is sequential with the billing process followed by payment process and then by execution of the functionality.

- **Cash sales:** This type of payment mechanism is used in day-to-day purchases, repair services, and on-line purchases. The user selects the service or the product, the cashier totals the amount and the user pays the amount to

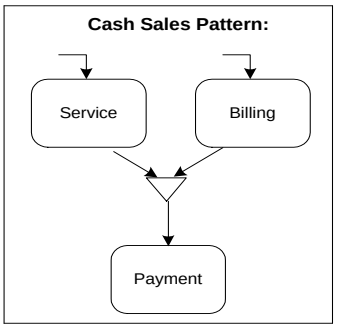

Fig. 11. Payment: Cash Sales

the cashier. In this case, the billing and execution may be carried out at the same time (order is not crucial). It results in the payment for the service as shown in figure 11.

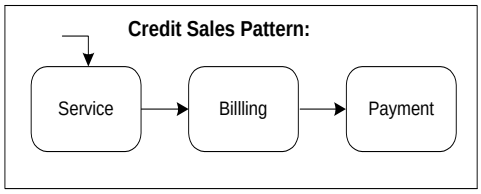

Fig. 12. Payment afterwards: credit sales

– **Credit sales:** This type of payment mechanism is used mainly in business to business transactions. The business customer sends an purchase order, the order is executed and the invoice is sent to the customer with the due date set depending on the credit terms. In this case, the function is executed, the user is then billed and the payment process is performed at a later time as in figure 12.

4 Issues raised by using patterns

We are interested in the development of business applications using pre-defined patterns as well as in the effects of incorporating non-functional requirements into payment mechanisms. Our approach to service composition using patterns raises a number of issues.

4.1 Patterns repository

Ideally, we would like to store all the patterns in a repository, indexed on a number of classifications to enable efficient search and reuse. The questions raised include:

- What type of patterns would be the most useful for business applications developers? The types of patterns vary between organisational patterns, analysis patterns, design patterns, process patterns etc. There is a need to identify what kind of patterns would be suitable for composite e-services.

- How could the patterns be derived from well-known situations? It will be necessary to identify well-known business process models and to derive patterns from these. A lot of work has been carried out to derive patterns for various problem domains and represent them as pattern catalogs see (http://www.hillside.net/patterns/onlinepatterncatalog.htm.)

- How should the patterns be classified in the repository? It might be that individual patterns belong to more than one category. The UDDI programming interface (API) defines a consistent way for businesses to add any number of classifications to their business registrations (UDDI.org 2001). The classifications used include category codes such as NAIC, UN/SPC, SIC codes, that are widely used to classify businesses, industries, and product categories as well as other classifications designate geographic information, or membership in a given organization such as industry codes, product codes, geography codes and business identification codes. Gamma et al. (1994) organise design patterns according to the purpose and scope of the patterns. They also suggest other ways of classification such as grouping the patterns that are mostly used together, or alternative patterns or patterns with similar outcomes. We also need to consider the ontologies approach to classification.

- How should the patterns that are specific to e-service composition be described? It is likely that some patterns might be specific to service composition. Appleton (2000) states that the following essential elements should be clearly recognizable upon reading a pattern: name, problem, context, forces, solution, examples, resulting context, rationale, related patterns and known uses. We need to consider the aspects that are the most important for the patterns to be reused in e-service composition and describe them accordingly.

These questions highlight the issue of what type of patterns will be included in the repository for service composition and how to describe and classify the patterns.

4.2 Pattern instantiations

We are interested in using a top-down approach to service composition, capturing business goals in terms of established patterns. The issues raised in instantiating generic patterns into specific or domain dependent patterns include:

– What is the right level of abstraction for the developer? The generic patterns could be the building blocks in the repository. The patterns should also describe the specific functionality that is supported such as holiday planning or internet sales.

– How can the gap between business goals and the initial generic pattern be described? In this case, we should consider the work on multi-criteria decision making by Corner et al. (2001). The dynamic process of structuring a decision problem involves the specification of options, attributes for evaluating options and states of nature that may occur, with repeated cycling back in the process to revise or augment the structure. Corner et al. (2001) advocate a dynamic interaction between criteria and alternatives as a decision-maker understands his preferences and expands the set of alternatives.

– How can the gap between available services and the composer's needs be measured? It is necessary to identify the mismatch between the descriptions of available services and processes in the patterns. To find a service that match exactly with user's requirements will be very difficult. We should consider services that match the functionality but still need adaptation.

– What will be the trigger to stop instantiating the pattern? All tasks might not be performed by e-services. The business may want to use in-house functionality that is not implemented as an e-service. User interaction is needed to indicate the functionality that he/she wants to outsource.

A lot of work still needs to be done on bridging the gap between user requirements and patterns as well as the gap between patterns and service descriptions.

4.3 Non-functional requirements (NFR) patterns

Service composition is much more than functional composition. We should be able to represent other non-functional requirements as patterns and incorporate them into decision making. We have attempted to illustrate the use of payment mechanism patterns as part of service composition. A number of issues are raised in terms of payment mechanisms patterns and in general non-functional requirements patterns. They include

– How does the payment mechanism for components affect the composite service's payment mechanism? The selection of a payment mechanism for the composite service is not directly related to the payment mechanisms of its components. However, we observe that the businesses are likely to ask for payment in advance for their services and are also likely to ask for credit terms to pay for their components.

– What are the influencing factors in determining the appropriate payment mechanisms for composite services? Some of the factors might be the type

of industry, the type of business, and the type of customer.

- Is the payment mechanism inseparable from the service? There are services that are free of charge, hence it seems that the payment mechanism can be separated from the service. Even though these services do not offer service guarantees, they could be used reliably if there are third party recommendations and review.

- How can patterns be used to represent non-functional requirements? The normal specification mechanisms focus on functional issues. Non-functional requirements include the methods of charging and payment, constraints on temporal and spatial availability, service quality, trust and quality (O'Sullivan et al. 2002). Most of them are hard to define and quantify and they are also relative to the user.

- How could the interactions between NFR patterns be represented? One non-functional requirement could impact on the other requirements. We must take into account the interdependencies and trade-offs between NFR patterns.

Non-functional requirements plays a crucial part in decision making process for service composition. Consideration should be given on how to represent these non-functional requirements as patterns.

5 Conclusion

We believe that patterns combined with the domain knowledge could be used to develop composite services in a systematic way. Our objective is to take the user through different levels of abstraction from generic to specific, resulting in a concrete business application using e-services. Within this paper, we have attempted to describe how generic patterns could be instantiated into specific patterns with the educational service example. We have also highlighted the importance of non-functional requirements for service composition and described how they can be represented in patterns using the payment mechanisms example. We have also raised a number of questions regarding patterns and in particular the use of patterns to represent non-functional requirements. The nature of patterns and the classification of patterns in the repository will play a major part in service composition. We also need to identify how to configure a match between processes in the model and the available e-services. To develop a pattern repository that could be used to compose many types of business applications, these issues must be carefully considered.

References

Alexander, C. (1979), *The Timeless way of Building*, Oxford University Press, New York.

Appleton, B. (2000), 'Patterns and Software: Essential Concepts and Terminology'. http://www.enteract.com/~bradapp/docs/patterns-intro.html#PatternElements accessed on 7 Mar 2002.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. & Stal, M. (1996), *Pattern-Oriented Software Architecture - A System of Patterns*, Wiley and Sons Ltd., USA.

Colan, M. (2001), 'An overview of Web Services'. http://www-106.ibm.com/developerworks/webservices/ accessed on 3 Sep 2001.

Corner, J., Buchanan, J. & Henig, M. (2001), 'Dynamic Decision Problem Structuring', *Journal of Multi-Criteria Decision Analysis* **10**, 129–141.

Edmond, D. & ter Hofstede, A. (2000), Service composition for electronic commerce, *in* 'Proceedings of the Pacific Asia Conference on Information Systems(PACIS-2000)', Hong Kong.

Fensel, D. & Bussler, C. (2002), 'The Web Service Modeling Framework (WSMF)'. http://www.cs.vu.nl/~dieter/wsmf/wsmf.paper.pdf accessed on 8 Mar 2002.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1994), *Design Patterns: Elements of Reusable Object-Oriented Software*, Professional Computing Series, Addison-Wesley, USA.

Gross, D. & Yu, E. (2001), 'From Non-Functional requirements to Design through Patterns', *Requirement Engineering* **8**, 18–36.

IBM (2002), 'developerworks: Patterns for e-business'. http://www-106.ibm.com/developerworks/patterns/ accessed on 28 Feb 2002.

Kotov, V. (2001), Towards Service-Centric System Organization, Technical Report HPL-2001-54, Hewlett-Packard. http://www.hpl.hp.com/techreports/2001/HPL-2001-54.html accessed on 3 Sep 2001.

Lord, J. G. J. (2001), 'Facilitating the application development process using the ibm patterns for e-business'. http://www-106.ibm.com/developerworks/patterns/guidelines/lord.pdf accessed on 12 Mar 2002.

O'Sullivan, J., Edmond, D. & ter Hofstede, A. (2002), 'What's in a service? Towards accurate description of non-functional service properties', *Distributed and Parallel Databases Journal - Special Issue on E-Services (to appear)*.

Piccinelli, G. (1999), Service Provision and Composition in Virtual Business Communities, Technical Report HPL-1999-84, Hewlett-Packard. http://www.hpl.hp.com/techreports/1999/HPL-1999-84.html accessed on 23 Jun 2001.

Piccinelli, G. & Mokrushin, L. (2001), Dynamic Service Aggregration in Electronic Marketplaces, Technical Report HPL-2001-31, Hewlett-Packard. http://www.hpl.hp.com/techreports/2001/HPL-2001-31.html accessed on 23 Jul 2001.

Salingaros, N. A. (2000), 'The structure of pattern languages', *Architectural Research Quarterly* **4**, 149–161.

UDDI.org (2001), 'UDDI Version 2.0 API Specification'. http://www.uddi.org/pubs/ProgrammersAPI-V2.00-Open-20010608.pdf accessed on 5 Sep 2001.

Yang, J. & Papazoglou, M. P. (2002), Web Components: A Substrate for Web Service Reuse and Composition, *in* 'Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE'02), May 27 - 31, 2002', Toronto, Canada.