FACULTY OF BUSINESS AND LAW

School of
Management Information Systems

Leaders in Business Computing

School Working Papers Series 2002

SWP 2002/03

Exploring Reuse Spaces of Web Services and Contents

Authors:
Jacob L. Cybulski
Tanya Linden
Pradipta K. Sarkar

URL:
http://www.deakin.edu.au/mis/research/working_paper.htm

Deakin University
School of Management Information Systems
Working Papers 3/2002

EXPLORING REUSE SPACES OF WEB SERVICES AND CONTENTS

Jacob L. Cybulski[1], Tanya Linden[2] and Pradipta K. Sarkar[1]

[1] School of Management Information Systems
Faculty of Business and Law
Deakin University
Burwood, Vic 3125, Australia
Email: jcybulski@acm.org

[2] Department of Information Systems
Faculty of Science
University of Melbourne
Vic 3010, Australia
Email: tanyal@unimelb.edu.au

ABSTRACT

Development of commercial web systems is laborious, lengthy and costly. This is partly due to the fact that the methods of their development can hardly cope with the complexity of provided services. Such services may need to be distributed and collaborative, require sophisticated software architecture, be rich in form, content and interactivity, and have a wide range of potentially casual users. To improve this situation, the authors propose a reuse space analysis (RSA) approach to web development. Our approach focuses on capturing domain and development experience of all system stakeholders and subsequently using this experience in making informed design and reuse decisions across the development life cycle. As a result we managed to seamlessly integrate design and reuse processes, and reaching the balance between the development cost, system function and its quality.

1. INTRODUCTION

Commercial web products, such as web-based e-commerce sites (e.g. Amazon.com), integrate software, information and web services into a cohesive package, designed to support products and their brand, customers and vendors, and business activities along the supply chain. Examples of web services include providing product information, product finding and selection, ordering and purchasing, billing and paying, packaging and delivery, rating and ranking of products, customer testimonials and reviews, promotions and advertising. Integration of such diverse services commonly leads to very complex web systems, which may consist of great many subsystems and modules, have sophisticated structure and interdependencies, of which development and maintenance may necessitate the use of specialised software tools and large repositories of components. Development of web products may also require expertise from distinct fields of business, publishing and marketing, video production and art design, software development, operating system configuration and hardware interfacing, thus rendering their development laborious, lengthy and usually very costly.

In view of these impediments, whether functional, informational or organisational, we consider web design a significant challenge, and like some others, e.g. Johnson and Nemetz (Johnson and Nemetz 1998), we advocate and undertake further research to develop more effective web design practices, some of which are based on new principles and methods.

To deal with the complexity of web products, special design methods and software to support them have already been developed. The majority of web design models have been adopted from software design, hence such models support the notions of objects or entities, their properties, relationships and constraints (Brodie 1984). In web design, there is also a need to address aspects, such as navigation,

user-interface, and content storage. These issues have been incorporated in some of the multimedia and hypertext models, e.g. Dexter Hypertext Reference Model (Halasz and Schwartz 1994) and Amsterdam Hypermedia Model (Hardman, Bulterman*, et al.* 1994), Object-Oriented Hypermedia Design Method (OOHDM) (Schwabe and Rossi 1995), ZyX model (Boll and Klas 1999, Boll, Klas*, et al.* 1999), and visual composition models (Mey, Breiteneder*, et al.* 1992, Mey and Gibbs 1993).

Since web development is commonly based on easily identifiable multimedia components, such as images, video or audio clips, and text fragments, it is thus eminently suited a reuse-based process. Research in software reuse shows positive influence of component reuse on software quality, productivity and product delivery time (Lim 1994, Tracz 1988). Although there is as yet no convincing proof that the same applies to reuse of web-media, some researchers believe that component reuse has a significant impact on web and multimedia development (Garzotto, Malinettit*, et al.* 1996). Research in multimedia design shows that developers may also benefit from reuse of designs (Boll and Klas 1999, Rossi, Schwabe*, et al.* 1997). A number of multimedia design methodologies address the issue of reusability and enforce reuse practices specifically during design. For example, OOHDM (Object-Oriented Hypermedia Design Model) facilitates systematic reuse of multimedia components and design structures across hypermedia development process (Schwabe and Rossi 1995). Rossi and colleagues (Rossi, Garrido*, et al.* 1996) propose a number of hypermedia patterns that can be reused to guide the process of hypermedia design. The ZyX model (Boll and Klas 1999, Boll, Klas*, et al.* 1999) addresses the issues of reusability by providing methods of identifying and selecting reusable multimedia fragments, which may include information contents, structure and layout. Finally, de May describes an interactive method of multimedia construction by visual composition of components, thus promoting their sharing and their reuse (Mey, Breiteneder*, et al.* 1992, Mey and Gibbs 1993).

To make reuse practices acceptable to web developers, they need to be well integrated with technical development activities across the product life cycle, i.e. gathering and refinement of requirements (for content, presentation and services), and their subsequent design, implementation and deployment. Non-technical activities, such as project planning, producing multimedia artwork or designing a business supply chain, should also be supported by the adopted development method. Considering the diversity of activities associated with web development, we decided to turn our attention to supporting reuse in the creative process of design deliberations, which in our view is common to all development activities, i.e. those associate with technical, artistic and business aspects of web products. We call such an approach *frugal web development*, as it regards all aspects of the new system development with a special focus on the reuse of the available components and designs, thus providing economy in the use and appropriation of web resources.

2. DESIGN DELIBERATIONS FOR WEB DESIGN

Web developers design various aspects of future systems using a number of different representations. For example, screen shots and sketches are used to show user interface layouts, storyboards and scenarios illustrate system interaction with the user, collaboration and navigation charts depict communications and transfers between web nodes, entity-relationship and class diagrams represent the structure of information repository, state charts and Petri nets are used to specify the communication protocols adopted by the communicating web agents, etc. While producing such design artefacts, developer teams commonly deliberate their major design decisions. Design deliberations reflect the issues associated with the proposed designs, problems encountered and the possible solutions to these problems (Conklin and Begeman 1987). Recording and analysing design deliberations can assist developers making informed decisions about various aspects of developed products. In web development this could be helpful in selecting web services to meet user requirements, deciding on the composition method of web information content, on using the appropriate visual and interactive features of web pages, or planning the navigational paths throughout the presented information.

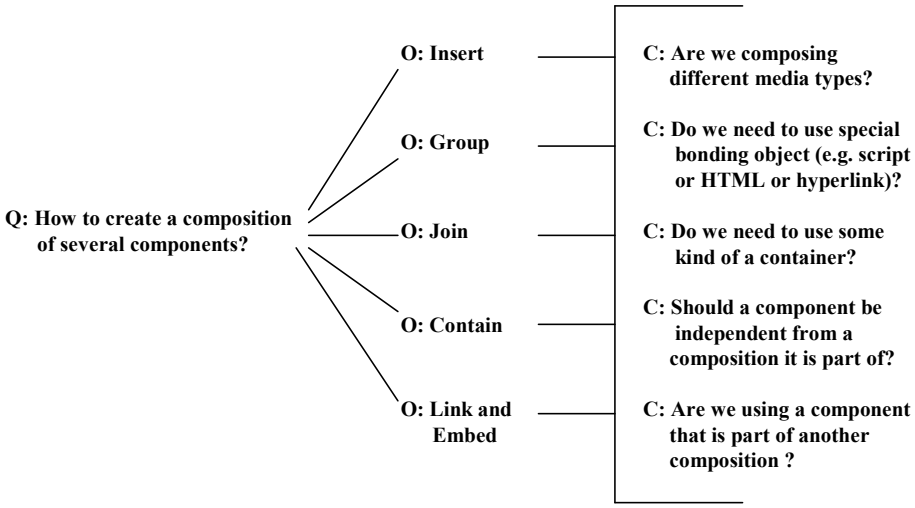

Figure 1: Alternative design options

Several methods and systems have been developed to capture design deliberations, e.g. gIBIS (Conklin and Begeman 1987), SIBYL (Lee 1990) and QOC (MacLean and McKerlie 1995). Such approaches differ in their ways of capturing design decisions, their diagrammatic representation, and in the use of their basic terminology. However, they are remarkably similar in using an obstacle in a design process as a starting point for evaluating alternative design options to support developers' decision making.

- gIBIS (graphical Issue Based Information Systems) and a more generic IBIS (Conklin 2000) allow capturing design decisions and their justification in the process of group discussions. In the centre of discussions there may exist some key issues that in IBIS are formulated as questions. Different ideas of answering these questions, and thus resolving the central issues, are then noted and justified. A network of questions, ideas and justifications help the discussion participants to freely explore alternatives in the design process and to reach consensus, leading to a design decision. The focus of the IBIS process is on *conflict resolution*.
- SIBYL is a distributed qualitative decision management system, which aims at documenting a decision-making process with the use of a special notation, Decision Representation Language (DRL). SIBYL provides means to record decisions and supporting evidence, including alternatives and constraints, and other relevant information for evaluating those alternatives. The SIBYL's focus is on a *decision making process*.
- QOC (Questions, Options, Criteria) is similar to gIBIS and SYBIL in its claim of resolving hard design issues. QOC poses *questions*, which reflect key aspects of a design process. In web design, for example, such a question may relate to the method of composing several multimedia artefacts (see Figure 1). For each question a number of alternative design *options* are proposed, thus providing a list of possible answers to the design question, e.g. the composition methods may include insertion, grouping, joining, containment or linking multimedia artefacts. *Criteria* are requirements of which combinations are considered while choosing one option over another, e.g. criteria may probe the source and type of composed artefacts, or their use after the composition has been formed. The QOC method can be used to navigate an entire space of design questions and answers. MacLean and McKerlie (1995) use such design spaces as an explicit representation of design alternatives and they utilise QOC to seek reasons for choosing between them. The focus of QOC and the associated design spaces is on the *selection of design options*.

In our research into web design, we adopted the QOC approach to design deliberations, which allows us to sharply focus on evaluating different design options in web development. To support reuse in this design process, we formed a number of useful analogies, in which we identify a space of design

options with a reuse repository, design deliberations with the process of component search, design decisions with component retrieval, and recording design rationale with component classification. We call such an approach a *reuse-space analysis* (RSA). As proposed, RSA satisfies the requirements of a typical reuse method with its ability of component classification and search, storage and retrieval, and their selection for integration into products (Cybulski and Linden 1999).

In the following sections we will provide an example of the RSA process, leading from requirements identification to components selection for web construction. The example is based on a case study of development issues in six web enabled payroll systems, which we conducted across organisations providing outsourced payroll services in Melbourne (Australia).

3. FROM STAKEHOLDERS' CONCERNS TO REQUIREMENTS

Requirements acquisition is commonly the starting point for any large systems development. System *requirements* find their source in the *business needs* of an organisation. While aligning such needs with requirements, we find that many requirements can be reused from other systems in the same application domain. Should requirements reusability be successful, this may also lead to the reuse of development work-products, which have been previously derived from these requirements. RSA, thus, advocates aligning business needs with reusable domain requirements, with a view to promote reusability at the earliest stage in web development.

In order to prompt this process, business needs are gathered from each *stakeholder* of the proposed system. Each need raises one or more issues related to the system form, structure, function or its use (questions), which can be addressed by numerous requirement alternatives (options), to be selected in satisfaction of stakeholders' concerns (criteria). Some of these requirements may be reusable, others may relate to the entire new system features and functions. Requirements and their selection criteria can be discovered in the normal course of *systems engineering* (SE) or by conducting *domain analysis* (DA). While QOC is used predominantly in systems engineering tasks, the RSA method also suggests employment of domain analysis, which is known to undertake a systematic study of a domain with the purpose of developing various reusable software artefacts, i.e. domain models, reusable architectures and software components.

One of the most notable domain analysis methods is FODA (Feature-Oriented Domain Analysis - Kang, Cohen, *et al.* 1990) and its extension, FORM (Feature-Oriented Reuse Method - Kang, Kim, *et al.* 1998). While strongly maintaining that the goal of the domain analysis is to facilitate reusability, the authors of FODA and FORM contend that multiple systems in a domain can be studied in order to derive a set of "commonalities" and "variations" in their contents, structure and functionality. The system properties being the source of these commonalities and variations are referred to as *features*. Once identified, features can later assist developers in determining reusable components and architectures suitable for the new system development. To distinguish between system requirements, they too can be classified according to the features discovered in domain analysis. Such features are then used to form the RSA criteria, and the feature values assist the selection of requirement options to refine business needs.

Combinations of features and constraints imposed on their values represent *stakeholder concerns*. According to Somerville and Kotonya (1996), a "concern" is a domain requirement, the compliance with which is critical to the success of the development process and thus, the system to be developed. Thus, concerns need to be considered while designing a system. In RSA, stakeholder concerns are of prime determinant of the selection of components for reuse.

Question: ***How to ensure that hours worked are entered correctly?***	**Criteria**	*Specified in terms of the person / organisation responsible for a specific activity in the process of generating payroll.*			
Options:	**Data Collection**	**Data Entry**	**Pre-Verification**	**Data Processing**	**Post-Validation**
The employer should supply legible and accurate timesheets.	Employer	Payroll		Payroll	
The employee should verify that the payment received matches the submitted timesheets.				Payroll	Employee
The employer should verify data entered by payroll people.		Payroll	Employer	Payroll	
The employer should enter the timesheet data and verify it to be correct.	Employee	Employer	Employer	Payroll	
The employee should enter the timesheet data and the employer should verify it to be correct.	Employee	Employee	Employer	Payroll	
The employer should provide some data redundancy to enable its automatic verification by payroll people.		Employer	Employer Payroll	Payroll	
The payroll provider should supply a clocking device to automate the collection and verification of hours worked.	Payroll	Payroll	Payroll	Payroll	

Table 1 - Requirements identification

To illustrate the process described thus far, let us explore a fragment of a case study that we've conducted to investigate six web-based payroll systems in Melbourne (Australia). In the application domain of web-enabled payroll, the stakeholders are employees, employers, and payroll (which usually also includes IT support), who are involved in the collection of hours worked by employee, entering this data into electronic form, verification of payroll data, running of the payroll, and finally validation of the calculated pay. One of many business needs identified in the course of our study (see Table 1) was formulated as the following RSA question: "How to ensure that hours worked are entered correctly?". This business need can be satisfied in many possible ways, each resulting in a different requirement that could be imposed on the web-based payroll system. Alternate requirements may result in systems ranging from a totally manual collection and verification of timesheet data to a fully automated system monitored by the payroll organisation. Our analysis of this particular business need and its requirement alternatives resulted in a set of domain features, related to the stakeholder responsibility for the completion of a task in the payroll workflow. We used these features as RSA criteria to classify the collected requirement options. The resulting *decision table* can subsequently be used to align business needs with one or more requirements. The selection depends on the specific needs of an individual organisation and the concerns of its various stakeholders.

For example, we found in one of the payroll organisations that the concerns of the employer relate to filling of timesheets by the employees themselves (the main motivator behind the adoption of their web-based applications), and the subsequent approval by the supervisor (employer). The payroll organisation's concerns include the data entry at the “source”, and the arrival of timesheet data that has already been verified. These concerns were then used to construct an RSA *query* consisting of specific feature values, which were then *matched* against the criteria, thus resulting in the selection of a requirement satisfying all of the relevant organisational concerns: "The employee should enter the timesheet data and the employer should verify it to be correct." (shown as a shaded requirement option

Question: ***How to guide payroll users during their data-entry tasks, in view of the need to conform to the rules of data integrity?***	**Criteria**	*Specified in terms of the domain features related to the user type, triggering event, and delivered information.*			
Options:	**Data Entry**	**Task Awareness**	**Trigger Event**	**Information Volume**	**Information Precision**
Display precise error messages.	Employee	Confident	Error	Low	Very High
Use context-sensitive on-line help.	Employee	Confident	Task	Medium	High
Walkthrough a demo program.	Employee	Confident	Training	High	Low
Use data entry wizards.	Employee	Naive	Session	Medium	High
Use hard-copy manual.	Employer Payroll	Expert Confident Naïve	Error Task Session Training	High	Very High
Learn integrity rules at training.	Employer Payroll	Expert Confident Naïve	Training	High	Medium

Table 2 - Requirement refinement into design options

in the decision Table 1). The selected requirement, hence, aligns with the business need of that specific organisation. It should also be noted that as the majority of investigated organisations offered several payroll products and services to their clients, the selected requirements was in fact reused across many of these offerings.

As we can see from the presented example, the RSA method and its decision tables can be used to align business needs with requirements, some of which could be reusable in an application domain, to address the concerns of stakeholders in a given organisation.

4. FROM REQUIREMENTS TO DESIGN COMPONENTS

The key to the successful refinement of requirements into design components is in the effective mapping of requirements into a space of software designs, of which a significant number should be reusable (Cybulski and Reed 1999). This is normally done manually by teams of computer analysts and designers. However, the process can be enhanced with the use of group decision support systems (Nunamaker, Briggs*, et al.* 1995) and information retrieval techniques (Castano and De Antonellis 1994, Frakes, Prieto-Diaz*, et al.* 1998, Prieto-Diaz and Freeman 1987). Both approaches call upon methods of classifying, searching, retrieving and selecting development information. While QOC and RSA provide such facilities in their approach to design process, it was shown in the previous section that RSA also emphasises explicit classification of alternatives to align business needs with reusable requirements. RSA also relies on the classification of designs and web components to identify reusable resources useful in requirements refinement. Table 2 illustrates this process.

Requirements can be refined in much the same fashion as discussed previously, i.e. by listing their features, classifying them in terms of feature values and then using a combination of these features values as selection criteria to determine the best refinement option. Alternatively, we can look for previously considered design options, which have been recorded in decisions tables that share some of the features with the selected requirement. For example, Table 1 shares a "Data Entry" feature Table 2, which attempts to resolve a design question "How to guide payroll users during their data-entry tasks,

Question: ***How to communicate error messages to users?***	**Criteria**	*Specified in terms of the domain features related to the media form and contents, the level of intrusion on user tasks, and volume and precision of communicated information.*			
Options:	**Media Form**	**Content**	**Intrusion**	**Volume**	**Precision**
Show pop-up window with a message.	Graphics Text	Instruction	High	Medium	High
Display messages in an error field.	Text	Warning	Low	Low	Very High
Sound an audible alert.	Beep	Attention	Very Low	Very Low	Low
Provide a voice over.	Voice	Instruction	Low	High	High
Flicker a menu bar.	Blink	Attention	Low	Very Low	Low
Permanently display a window to a live instructor / administrator.	Video	Dialogue	Medium	Very High	Medium

Table 3 - Selection of design components

in view of the need to conform to the rules of data integrity". Clearly, this design issue is of relevance to the requirement "The employee should enter the timesheet data and the employer should verify it to be correct" and thus to its direct business need "How to ensure that hours worked are entered correctly". In general, decision tables evaluating various design options, which intersect with decision tables representing alignment of requirements with business needs, are of relevance to the requirements refinement process.

Table 2 shows four of its six design options as compatible with the selected requirement, which is based on the value "Employee" of the "Data Entry" feature (it should be noted here that matching of feature values can be partial and fuzzy, which can both be supported with domain dictionaries and thesauri - Frakes, Prieto-Diaz*, et al.* 1998). Assessment of other selection criteria and their associated features, such as the required level of user confidence, the message triggering-event, or information volume and precision, can all be useful in making an informed reuse / design decision. Taking into consideration all of the criteria associated with a particular RSA table is in fact recommended, as the criteria listed are those previously found as discriminating between the available (and potentially reusable) alternatives. The highlighted choices in Table 2, i.e. the use of error messages and context-sensitive help, correspond to a design decision based on the user's high confidence and the requirement for a low-to-medium volume of information presented to the user. The process of design refinement can be applied iteratively, until such time that individual design decisions can easily map onto simple implementation options or they could associate reusable components, which could subsequently be integrated into a ready-made web solution. Table 3 illustrates the case when one of the previously selected design options, "How to communicate error messages to users", has been further refined into one of the reusable web components, i.e. the use of error fields to display messages.

In many situations, information contained in an RSA table is insufficient to make a high quality, informed design decision. In such a case, we suggest that a designer should turn to other, external, sources of information. However, should this also be the case that a design issue is pertinent to several systems in a given application domain, and so the specific decision making process needs to be frequently retraced, we then advise to package developers' experience in dealing with this particular domain issue. As a result of such packaging, we can create a repository of domain experience, similar to Basili's *experience base* (1994). Collected experience of expert designers working in a given application domain can be reused in new projects leading to a significant reduction of stakeholder concerns. In RSA, we package design decision experience in a form of patterns, considered by others as suitable for sharing and reusing expert knowledge (Buschman, Meunier*, et al.* 1996, Rising 1999).

5. SUMMARY

RSA provides a method of supporting web developers in a range of development activities leading from business needs analysis, through requirements identification and high-level design, down to the composition of reusable services and contents into a new web product.

Our method assists developers in making informed design decisions while planning the web product's form, contents, structure, function, aesthetics and usability. At the same time, it provides means of identifying reusable web resources that could be used in the design process, thus leading to savings in development time and cost, increasing developer's productivity and raising the quality of the resulting web products and services.

It also promotes the use of domain analysis to establish a framework for capturing and navigating a web design space. Such a design space is subsequently used to collect and classify requirements, designs and reusable web components, so that they could be easily found and reused in the process of solving design problems during the course of web systems engineering.

RSA suggests packaging developers experience in dealing with the frequently encountered web design problems into design patterns. Such patterns clarify the problem context, the choice of the available options, and some of the difficulties which may impede the problem resolution.

To date, our case studies focused on the employment of the RSA method in conducting domain analysis, creation of the web design space, and capturing and evaluating design patterns in web application development. Our future empirical work will centre on use of RSA to support reusability of web services and contents in systems engineering tasks, i.e. requirements identification and refinement, and the component-based design of web applications.

6. REFERENCES

Basili, V.R., C. G.;, and H.D. Rombach (1994): "The Experience Factory", in *Encyclopedia of Software Engineering*. John Wiley & Sons, Inc. p. 469-476.

Boll, S. and W. Klas (1999): "ZYX - A Semantic Model For Multimedia Documents And Presentations". in *8th IFIP Conference on Data Semantics (DS-8): "Semantic Issues in Multimedia Systems"*. Rotorua, New Zealand.

Boll, S., W. Klas, and U. Wstermann (1999): *A Comparison of Multimedia Document Models Concerning Advanced Requirements*, Technical Ulmer Informatik-Berichte Nr 99-01, Computer Science Department, University of Ulm: Ulm, Germany.

Brodie, M.L. (1984): "On the Development of Data Models", in *On Conceptual Modelling*, M.L. Brodie, J. Mylopoulos, and J.W. Schmidt (Editors). Springer-Verlag: New-York. p. 19-47.

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

Castano, S. and V. De Antonellis (1994): "The F3 Reuse Environment for Requirements Engineering". *ACM SIGSOFT Software Engineering Notes*. **19**(3): p. 62-65.

Conklin, J. (2000): *The IBIS Manual: A Short Course in IBIS Methodology*, Working Paper, Group Decision Support Systems Inc.

Conklin, J. and M.L. Begeman (1987): "gIBIS: A hypertext tool for team design deliberation". in *Hypertext'87*. Chapel Hill, NC: ACM, p. 247-251.

Cybulski, J.L. and T. Linden (1999): "Composing Multimedia Artefacts for Reuse", in *Pattern Languages of Program Design Vol 4*, N. Harris, B. Foote, and H. Rohnert (Editors). Addison-Wesley Longman. p. 461-488, ch 21.

Cybulski, J.L. and K. Reed (1999): "Automating Requirements Refinement with Cross-Domain Requirements Classification". *Australian Journal of Information Systems*(Special Issue on Requirements Engineering): p. 131-145.

Frakes, W., R. Prieto-Diaz, and C. Fox (1998): "DARE: domain analysis and reuse environment". *Annals of Software Engineering*. **5**: p. 125-141.

Garzotto, F., L. Malinettit, and P. Paolini (1996): "Information Reuse in Hypermedia Applications". in *The seventh ACM conference on Hypertext*. Bethesda, MD USA, p. 93-104.

Halasz, F. and M. Schwartz (1994): "The Dexter Hypertext Reference Model". *Communications of the ACM*. **37**(2): p. 30-39.

Hardman, L., D.C.A. Bulterman, and G.v. Rossum (1994): "The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model". *Communications of the ACM*. **37**(2): p. 50-63.

Johnson, P. and F. Nemetz (1998): "Towards Principles for the Design and Evaluation of Multimedia Systems". in *Human - Computer Interaction (HCI'98)*. Sheffield, England: Springer-Verlag.

Kang, K., S. Cohen, J. Hess, W. Novak, and S. Peterson (1990): *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie-Mello University.

Kang, K.C., S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh (1998): "FORM: a feature-oriented reuse method with domain-specific reference architectures". *Annals of Software Engineering*. **5**: p. 143-168.

Lee, J. (1990): "SIBYL: A Qualitative Decision Management System", in *Artificial Intelligence at MIT Expanding Frontiers*, P.H. Winston and S.A. Shellard (Editors). The MIT Press: Cambridge, Massachusetts. p. 104-133.

Lim, W.C. (1994): "Effects of Reuse on Quality, Productivity, and Economics". *IEEE Software*. **11**(5): p. 23-30.

MacLean, A. and D. McKerlie (1995): "Design Space Analysis and Use-Representation", in *Scenario-Based Design: Envisioning Work and Technology in System Development*, J.M. Carroll (Editor). Wiley: New-York. p. 183-207.

Mey, V.d., C. Breiteneder, L. Dami, S. Gibbs, and D. Tsichritzis (1992): "Visual Composition and Multimedia". in *Eurographics '92*. Cambridge, UK: Blackwell Publishers, p. 9-22.

Mey, V.d. and S. Gibbs (1993): "A Multimedia Component Kit: Experiences With Visual Composition Of Applications". in *Multimedia '93*. Anaheim, California, USA, p. 291-300.

Nunamaker, J.F.J., R.O. Briggs, and D.D. Mittleman (1995): "Electronic Meeting Systems: Ten Years of Lessons Learned", in *Groupware: Technology and Applications*, D. Coleman, and Khanna, R. (Editor). Prentice-Hall. p. 149-193.

Prieto-Diaz, R. and P. Freeman (1987): "Classifying Software for Reusabililty". *IEEE Software*. **4**(1): p. 6-16.

Rising, L. (1999): "Patterns: A Way to Reuse Expertise". *IEEE Communications Magazine*. **37**(4).

Rossi, G., A. Garrido, and S. Carvalho (1996): "Design Patterns for Object-Oriented Hypermedia Applications", in *Pattern Languages of Program Design*, J.M. Vlissides, J.O. Coplien, and N.L. Kerth (Editors). Addison-Wesley: Reading, MA. p. 177-191.

Rossi, G., D. Schwabe, and A. Garrido (1997): "Design Reuse in Hypermedia Applications Development". in *8th ACM Conference on Hypertext (Hypertext'97)*. Southampton, England.

Schwabe, D. and G. Rossi (1995): "The Object-Oriented Hypermedia Design Model". *Communications of the ACM*. **38**(8): p. 45-46.

Tracz, W. (1988): "Software reuse: motivators and inhibitors", in *Software Reuse: Emerging Technology*, W. Tracz (Editor). Computer Society Press: Washington, D.C. p. 62-67.