Sharing and Reuse of Web Developers Experience

Jacob L. Cybulski and Pradipta K. Sarkar
School of Information Systems
Faculty of Business and Law
Deakin University
221 Burwood Highway
Burwood, Vic 3125, Australia
Email: jlcybuls@deakin.edu.au

Tanya Linden
Department of Information Systems
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. While web development methodologies are slowly emerging from research methodologies, their adoption is far from wide-spread. To improve this situation, the authors propose an approach based on the observed practices of professional web developers, who openly share and reuse their web development experience, while guarding their development work-products. To this end, we suggest to enhance web development methods, by collecting problem-solving experience of web developers, packaging it into a reusable form, and providing a navigable decision structure assisting developers in identifying proven expert solutions suitable for a given problem context.

Keywords

Web development, experience reuse, patterns, design deliberations

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 (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 & Schwartz, 1994) and Amsterdam Hypermedia Model (Hardman, Bulterman, & Rossum, 1994), Object-Oriented Hypermedia Design Method (OOHDM) (Schwabe & Rossi, 1995), ZyX model (Boll & Klas, 1999; Boll, Klas, & Wstermann, 1999), and visual composition models (de Mey, Breiteneder, Dami, Gibbs, & Tsichritzis, 1992; de Mey & Gibbs, 1993).

Since web development is commonly based on easily identifiable multimedia components, such as images, video or audio clips, and text fragments, it *seems* that a reuse-based process would be eminently suited for such development. 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, & Paolini, 1996). Research in multimedia design shows that developers may also benefit from reuse of designs (Boll & Klas, 1999; Rossi, Schwabe, & Garrido, 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 & Rossi, 1995). The ZyX model (Boll & Klas, 1999; Boll 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 (de Mey et al., 1992; de Mey & Gibbs, 1993).

Modern web and multimedia development products, such as Macromedia Dreamweaver and Fireworks[1] or Adobe GoLive and Photoshop[2], are also capable of supporting many reuse tasks. They assist developers in reusing multimedia content (via media libraries) and presentation (via templates), web system functions (via script libraries and history mechanism) and services (via server-side modules), and the subsequent multimedia asset management (via version tracking). Many less-technical activities, such as project documentation (with documentation editors), project management (via user management and status tracking), and web site management (via site deployment and archiving) are also supported by these tools, and can be readily incorporated into the selected multimedia development methods (Lam, Jones, & Britton, 1998).

It was this conviction of reuse importance to web development, as shared by many web and multimedia researchers, methodology proponents and tool makers, which motivated our research of the seemingly common reuse practices. Having interviewed 7 independent web / multimedia developers and investigating 14 separate projects, it came to us as a shock to find out that in spite of numerous publications devoted to reuse of web development work-products, such practices are still virtually non-existent. Analysis of the collected empirical data led us to a number of interesting observations. Firstly, professional web developers have a very strong sense of ownership for the creative web contents, thus, they actively oppose sharing their work products. Secondly, they are rewarded by their clients to deliver products, which are unique in contents and function, and hence, web developers tend to resist reuse that could potentially lead to the perception of sameness and mediocrity. And thirdly, the majority of the investigated web development projects were “all-inclusive”, and so, they were scoped in a way to offer very little opportunities for cross-project sharing and reuse. Although, it is quite likely that developers’ resistance to reuse, as indicated in the preceding three points, could be overcome by increasing their awareness of the potential benefits of reusability or by training them in the effective use of reuse methods (Frakes & Fox, 1995), we believe that the practitioners’ attitudes were quite well motivated. In fact, while openly opposed to sharing products of their labour, we found web developers to be seeking methods and opportunities of sharing, reusing and disseminating their development experience. This realisation had a major impact on our understanding of reuse practices in web and multimedia development and became the main focus of our ensuing research.

Our initial study of 7 web developers was subsequently followed by a series of 6 case studies of web-enabled Human Resources and Payroll systems. These case studies allowed us to investigate a range of typical activities associated with web development and developers’ experience in dealing with common problems in such development. In our work to date, we have focussed mainly on development of methods suitable for capturing and reusing this experience across a common application domain. Our method of choice for this task was the use of “experience patterns” (Sarkar & Cybulski, 2002b), i.e. packages of problem-solving experience that describe typical solutions to commonly occurring problems in a given domain, and which allow problem-solvers to focus on the more creative developmental aspects (Alexander, Ishikawa, Silverstein, & Jacobson, 1977; Appleton, 2000; Gamma, Helm, Johnson, & Vlissides, 1995). In our research we have demonstrated experience patterns to be extremely useful in two important facets of web development, i.e. in aligning system requirements with stakeholder concerns (Sarkar & Cybulski, 2002a) and then in selecting the right multimedia design artefact to fulfil a specific user requirement (Cybulski & Linden, 2000). Where patterns fail in the multimedia design process is to provide an overarching structure for deliberation and selection of appropriate patterns from the entire experience space to assist in solving the problem at hand.

1 See www.macromedia.com (Accessed: June 2003).
2 See www.adobe.com (Accessed: June 2003)

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, and state charts and Petri nets are used to specify the communication protocols adopted by the communicating web agents. While producing such design artefacts, developer teams commonly deliberate their major design decisions. Design deliberations reflect developers' experience in dealing with issues associated with the proposed designs, problems encountered and the possible solutions to these problems (Conklin & 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.

Several methods and systems have been developed to capture design deliberations, e.g. gIBIS (Conklin & Begeman, 1987), SIBYL (Lee, 1990) and QOC (MacLean & 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. 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-like approach to design deliberations, which allows us to sharply focus on evaluating different design options, as captured and packaged into the experience patterns, in web development. To support capturing and reuse of developers' experience in this design process, we formed a number of useful analogies, in which we identify a space of design options with an experience base, design deliberations with the process of searching for the problem-solving experience, design decisions with the identification of the experience patterns from this repository, and recording design rationale with the classification of the recorded experience. We refer to our approach *experience-base analysis* (EBA), being subjective reflection of expert web developers on their past problem-solving activity, this being quite in contrast to MacLean and McKerlie's (1995) objective design space. As proposed, EBA also satisfies the requirements for the construction of an automated reuse repository and its ability of component classification and search, storage and retrieval, and their selection for integration into products (Cybulski & Linden, 1999).[3]

In the following sections we will provide an example of the EBA 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 Automation is not a priority at this stage but is possible in the future.

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. EBA, 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 EBA 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, in our case experience patterns.

One of the most notable domain analysis methods is FODA (Feature-Oriented Domain Analysis - Kang, Cohen, Hess, Novak, & Peterson, 1990) and its extension, FORM (Feature-Oriented Reuse Method - Kang 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 EBA 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 EBA, stakeholder concerns, their similarities and differences have been found to be prime determinants of the selection of requirements in the alignment process with business needs (Sarkar & Cybulski, 2002a).

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 EBA 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 EBA criteria to classify the collected requirement options. The resulting “decision table”, which reflects developers’ experience in dealing with this problem, 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 EBA “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 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.

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

As we can see from the presented example, the EBA method and its decision support-tabular structures 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.

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 & 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, & Mittleman, 1995) and in case of very large decision spaces also information retrieval techniques (Castano & De Antonellis, 1994; Frakes, Prieto-Diaz, & Fox, 1998; Prieto-Diaz & Freeman, 1987). Both approaches call upon methods of classifying, searching, retrieving and selecting development information.

While QOC and EBA provide such facilities in their approach to design process, it was shown in the previous section that EBA also emphasises explicit classification of alternatives to align business needs with reusable requirements. EBA also relies on the classification of designs and web components to identify reusable resources useful in requirements refinement. Table 2 illustrates this process.

Requirements are 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 with Table 2, which in turn collects developers' experience in resolving a design question "How to guide payroll users during their data-entry tasks, 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.

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 automated data input validator.	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

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 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 design decision. Taking into consideration all of the criteria associated with a particular EBA 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.

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

EXPERIENCE PATTERNS

In many situations, information contained in an EBA 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 EBA, we package design decision experience in a form of patterns, considered by others as suitable for sharing and reusing expert knowledge (Buschmann, Meunier, Sommerlad, & Stal, 1996; Rising, 1999).

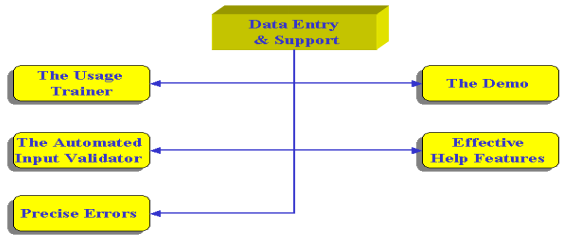
Pattern Group	***Data Entry & Support Concerns*** 
Pattern Name	***The Automated Input Validator***
Problem	***How to enforce the integrity of data entered by clients through the web-based interfaces?***
Context	Even though training provided by the payroll outsourcer or the HR department will familiarize client users with the various features of the web application, it may not, on its own, be able to ensure data integrity sufficiently, due to various reasons, which are described in the forces. Furthermore, many of the forces related to the previous pattern, The Input Validation Trainer, are also relevant in this context.
Forces	1. Clients may not be familiar with the data integrity rules of the application. 2. Service providers may not be present while clients are entering data. In such situations, complex data-entry procedures may force clients to consult the on-line Help of the application, which they may be averse to. 3. Clients may receive training in proper data-entry, but still enter records in violation of data integrity rules. Moreover, clients may attend training sessions, but the actual data-entry task might be assigned to someone employed as a casual/temporary staff. Thus, training may not guarantee that clients follow the rules. 4. The data entered by clients through the web application is directed to the database held by the payroll legacy system. In view of this, redundant or inaccurate data entered by clients may result in poor data management.
Solution	***The system shall embed data integrity and business rules into the application*** Data integrity rules could be enforced as non-functional requirements or constraints into the applications. There should be a mechanism that prevents the submission of the data entered incorrectly into the system. The constraints should prevent the record being committed to the database until the error is corrected. A related pattern is Unambiguous Format
Consequences	1. Embedding data integrity rules into the application will further enforce integrity by preventing data inaccuracy and redundancy. 2. The embedding of data integrity rules into the application requires significant development hours spent by the company's IT staff.
Known Uses	Most data processing systems, both web and non-web, have business and data integrity rules embedded into them. Examples of such patterns at work can be found in many of the internet banking and web-based flight reservation applications, where incorrect entry into certain fields, such as those for bank account number, credit card information, city and postcode, and dates, prevent the submission of an online form into the system. Integrity rules pertaining to the fields mentioned are usually embedded either into the interface or reside at the web server. For example, these rules will not allow the incorrect number of digits for the bank account number or a mismatch between city and postcode from being sent into the database. Similar types of measures are also undertaken in non-web applications.

Table 4 - Sample experience pattern

By domain-wide analysis of 6 Human Resource and Payroll organisations, their projects and their stakeholders, we have collected stakeholder concerns in respect to various problems with web-based payroll solutions, we then recorded and reconciled developer experience in dealing with these concerns, and in the end we have arrived at over 30 patterns detailing this experience. Table 4 shows a small group of patterns related to the design decisions explained in the previous sections (Cf. Table 1 and Table 2) and provides details of one of the patterns stored in our HR/Payroll experience base.

The pattern (The Automated Input Validator) addresses the specific concerns of web-system stakeholders in regards to the need of enforcing data integrity rules in situations when clients are permitted to submit their timesheet data. The pattern describes in detail the particular situation when the pattern applies, the nature of the problem, the forces which are known to cause the problem and which normally prevent the solution to be found, the solution as agreed upon by developers in this particular domain, and the impact of this proposed solution. Such patterns together with the navigable decision structure leading developers to the selection of the most appropriate solution for the problem at hand, form a comprehensive repository of web-developer experience spanning issues related to satisfying business problems, aligning them with system requirements, and refining these requirements into high-level (possibly reusable) designs.

To evaluate the pattern-based experience base, we have conducted a focus-group study in which a team of web developers - quite independent from the participants used in the formation of the experience base - used the collected patterns in solving a typical problem in designing an inter-organisational web-based payroll system. The developers found the patterns useful not only in identifying the problems in their case study and finding the necessary problem solutions but also useful in planning the overall approach to the system design. What was particularly satisfying was that in unison, the study participants, all being very experienced in developing web-based Human Resources systems, remarked on the fact that the patterns not only directly reflect their personal experience but also formalise it into a useable and accessible form. They also believed that such patterns would be excellent tool for dissemination of this collective experience and training new staff joining their team in the future.

SUMMARY

In this paper we have described an approach to representing, analysing and reusing experience of web-developers.

We proposed EBA method assisting web developers in making informed design decisions while planning the web product's form, contents, structure and function. In particular, EBA helps developers aligning web system requirements with business needs to minimise the potential stakeholder concerns. At the same time, the method 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.

The described approach promotes the use of domain analysis to establish a framework for capturing and navigating a web design and experience space. Such an experience 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.

An important part of the captured experience base is a collection of patterns representing the collective developers experience in dealing with the frequently encountered web design problems. Such patterns clarify the problem context, the choice of the available options, and some of the difficulties which may impede the problem resolution.

We believe that in the absence of mature web development methods, the outlined approach, being firmly grounded in the web development practice, can significantly enhance the quality and effectiveness of web development teams and their individual members.

The approach described in this paper could potentially contribute to development practices associated with other types of software products, e.g. engineering, control or corporate systems, which commonly exceed the size and sophistication of typical web-based information systems. The EBA applicability to these complex development domains, however, remains the focus of our future work.

REFERENCES

Alexander, C., Ishikawa, S., Silverstein, M., and Jacobson, M. (1977) *A pattern language*, Oxford University Press, New York.

Appleton, B. (2000) *Patterns and Software: Essential Concepts and Terminology* (Web www.enteract.com/~bradapp/docs/patterns-intro.html).

Basili, V. R., G.;, C., and Rombach, H. D. (1994) The Experience Factory, *Encyclopedia of Software Engineering*, (Vol. 2) John Wiley & Sons, Inc., 469-476.

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

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

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

Buschmann, F., Meunier, R., Sommerlad, P., and Stal, M. (1996) *Pattern-Oriented Software Architecture: A System of Patterns*, JohnWiley and Sons, Chichester.

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

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

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

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

Cybulski, J. L., and Linden, T. (2000) Composing Multimedia Artefacts for Reuse, in N. Harrison, B. Foote, and H. Rohnert (eds.), *Pattern Languages of Program Design*, (Vol. 4, Chapter 21) Addison-Wesley, Reading, MA.

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

de Mey, V., Breiteneder, C., Dami, L., Gibbs, S., and Tsichritzis, D. (1992) Visual Composition and Multimedia, *Proceedings of the Eurographics '92*, Cambridge, UK, 9-22.

de Mey, V., and Gibbs, S. (1993) A Multimedia Component Kit: Experiences With Visual Composition Of Applications, *Proceedings of the Multimedia '93*, Anaheim, California, USA, 291-300.

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

Frakes, W. B., and Fox, C. J. (1995) Sixteen questions about software reuse, *Communications of the ACM*, 38, 6, 75-87,112.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995) *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Readings, Massachusetts.

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

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

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

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

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

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

Lam, W., Jones, S., and Britton, C. (1998) Technology Transfer for Reuse: A Management Model and Process Improvement Framework, *Proceedings of the International Conference on Requirements Engineering, ICRE'98*, Colorado Springs, CO, USA, April 6-10, 233-240.

Lee, J. (1990) SIBYL: A Qualitative Decision Management System, in P. H. Winston and S. A. Shellard (eds.), *Artificial Intelligence at MIT Expanding Frontiers*, (Vol. 1) The MIT Press, Cambridge, Massachusetts, 104-133.

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

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

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

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

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

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

Sarkar, P. K., and Cybulski, J. L. (2002a) Alignment of Stakeholder Concerns with System Requirements, *Proceedings of the The Seventh Australian Workshop on Requirements Engineering, AWRE'2002*, Melbourne, Australia, 2-3 December

Sarkar, P. K., and Cybulski, J. L. (2002b) A Set of Patterns for the Web-based Interfaces of an Outsourced Payroll System, *Proceedings of the KoalaPLoP 2002 - the Third Asia-Pacific Conference on Pattern Languages of Programmes*, Melbourne, Australia, May

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

Sommerville, I., and Kotonya, G. (1996) Requirements Engineering With Viewpoints, *BCS/IEE Software Engineering Journal*, 1, 11, 2 - 26.

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

COPYRIGHT