

Separation of Concerns: a Web Application Architecture Framework

Xiaoying Kong¹ Li Liu² and David Lowe¹

¹Faculty of Engineering, University of Technology, Sydney,
P.O. Box 123 Broadway Sydney, NSW 2007, Australia

²Project Management Graduate Programme, Faculty of Engineering,
The University of Sydney, NSW 2006, Australia

Email: xiaoying.kong@uts.edu.au, l.liu@staff.usyd.edu.au, david.lowe@uts.edu.au

Abstract

Architecture frameworks have been extensively developed and described within the literature. These frameworks typically support and guide organisations during system planning, design, building, deployment and maintenance. Their main purpose is to provide clarity to the different modelling perspectives, abstractions, and domains of consideration within system development. In doing so they allow improved clarity with regard to the connections between the different models, and the selection of models that are most likely to capture salient features of the system. In this paper we present an Architectural Framework which takes into account the specific characteristics of web systems. The framework is based around a two dimensional matrix. One dimension separates the concerns of different participants of the web system into perspectives. The second dimension classifies each perspective into development abstractions: structure (what), behaviour (how), location (where) and pattern. The framework is illustrated through examples from the development of a commercial web application.

Keywords

Web architecture framework, architecture classification, web management, web modelling, web engineering, web information system, web architecture, web development tools

1. Introduction

Developing web systems is a complex endeavour that often requires the coordination of efforts across organizational and technical boundaries. People from different specializations and organizational units typically use their own technical languages and have unique values and norms. It is thus critical for organizations and people involved in a system development effort to understand the architecture of the system, defined as “*the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution*” (IEEE 2000). An Enterprise Architecture (EA) is often used to help with the understanding of the structure and functioning of the systems, and the roles and expectations of various stakeholders in a complex endeavour such as developing a large information system. It provides a map of the enterprise and is a route planner for business and technology change (Platt 2002). Using an Architecture Framework will speed up and simplify architecture development, ensure more complete coverage of the designed solution, and make certain that the architecture selected allows for future growth in response to the changing needs of the business (The Open Group 2003).

There are various existing Architecture Frameworks that establish terms and concepts pertaining to the content and use of architectural descriptions. Among others, the Zachman framework (Zachman 1987), 4+1 view model of architecture (Kruchten 1995) and Model Driven Architecture (MDA) (Object Management Group 2001) have received significant attention. Nevertheless, these Architecture Frameworks were developed for conventional enterprise information systems that do not have major web components. As we shall discuss below, as the complexity of web systems grows there is a need to develop an Architecture Framework to support and guide organizations during web system planning, design, building, deployment and maintenance.

This paper presents an Architecture Framework for web applications taking into account the unique characteristics of web systems. This framework is based around a two dimensional matrix. Each row of the matrix contains a set of architectures specific to a particular perspective – i.e. the concerns of a particular class of participants. Each perspective is then classified into different development abstractions: structure (what), behaviour (how), location (where) and pattern.

An Architecture Framework such as this supports developers in understanding the range of models and architectures which might be appropriate in developing Web applications or systems. The relevance of different perspectives and abstractions will vary depending upon the nature of the system or application being built, and it is important to be able to select those which best aid in supporting effective development. For example, a functionally rich Web-based work-flow system might be best developed by emphasising a functional perspective, whereas a content-rich catalogue system would probably emphasise information architectures. It would be highly unlikely to find a single system where it was appropriate to develop all perspectives and levels of abstraction.

In section 2 we consider the literature on Architecture Frameworks. Section 3 presents our Web Application Architecture Framework. Section 4 discusses findings and the implications of this framework. Finally in Section 5, the conclusions are drawn and future research directions are discussed.

2. Background

2.1 Existing Architecture Frameworks

Information technology related architectures for enterprise, information and data have evolved over the past 20 years. Among many Architecture Frameworks, Zachman's framework (Zachman 1987, Sowa et al. 1992) is widely acknowledged as the most comprehensive and sophisticated. Subsequently, the Zachman framework has become the basis for many variations of Architecture Frameworks. These frameworks support the observation that a system does not have a single architecture, but has a broad range of architectures representing different perspectives and different developmental abstractions.

The Zachman framework contains two dimensions. The first dimension describes the perspectives of stakeholders of the system, and includes the following viewpoints: ballpark; owner; designer; builder; subcontractor; functioning system; and description model. The second dimension presents six questions: what (data); how (function); where (network); who (people); when (time); and why (motivation). The two dimensions establish a matrix of 5x6 cells. Each cell describes a unique model, an architecture or a description. Each row represents a distinct and unique perspective (Zachman 1987).

Another popular framework is Kruchten's "4+1 view model of architecture" (Kruchten 1995) which consists of five views: logical; development; process; physical; and scenario. The logical view describes the services which the system should provide to its users. The process view concerns non-functional requirements such as performance and availability. The development view focuses on the actual software module organization within the software development environment. The physical view describes the mapping of the software onto the hardware. The elements in these four views are shown to work together seamlessly by the use of a small set of important scenarios. System engineers approach this "4+1" view model from the physical view, then the process view. End-users, customers, data specialists see it from the logical view. Project managers and software configuration staff use it from the development view. The five views are not fully independent and not all the views are always needed in supporting the development of specific software architectures (Kruchten 1995).

Similarly, Soni et al. (1995) classified software architectures for industrial applications into four categories: conceptual architectures; module interconnection architectures; execution architectures; and code architectures. Within each category the structure describes the system from a different perspective. The conceptual architecture describes the system in terms of major design elements and the relationship among them. The module interconnection architecture encompasses functional decomposition and layers. The execution architecture describes the dynamic structure of the system. The code architecture describes how the source code, binaries and libraries are organized in the development environment. The four architectures address different though inter-related engineering concerns.

The Object Management Group's Model Driven Architecture (MDA) (Object Management Group 2001) defines an approach to create models, refine models and generate code from models. Participants in a development process might use one of the three types of models: Computation Independent Models (CIM) that describes the business; Platform Independent Models (PIM) for architects and designers to describe the system architecture; Platform Specific Model (PSM) for developers and testers to generate code. Some aspects of the Zachman framework can be mapped into MDA (Frankel et al. 2003). In MDA, the choice of viewpoints is essentially a modelling choice.

Some other Architecture Frameworks such as the Federal Enterprise Architecture Framework (FEAF) (Chief Information Officer Council 2001), Command, Control, Computers, Communications, Intelligence, Surveillance, and Reconnaissance (C4ISR) (Department of Defence 1997), and the Treasury Enterprise Architecture framework (TEAF) (Chief Information Officer Council 2000) have been developed for government agencies. C4ISR claims to give comprehensive architectural guidance for all Department of Defence related domains. FEAF promotes shared development for US federal processes, interoperability, and shared information among US federal agencies and other government entities. TEAF provides an Architectural Framework that supports Treasury's business process in terms of work products (The Open Group 2003).

2.2 Web systems and Architectures

The nature of web systems is very different from the conventional software systems which the above frameworks are intended to support. At a technical level web systems typically: have tighter linkage between the business model and the technical architecture; have more pronounced open and modularised architectures; use technologies that change rapidly; demand effective information design and content management; place more emphasis on user interface; and place increased importance on quality attributes in mission critical applications that are directly accessed by external users (Lowe et al. 2001).

The existing Architecture Frameworks do not provide a clear pathway for addressing these characteristics. For example, the Zachman Framework matches the concrete entities, processes, locations, people, times, and purposes of the real world to the abstract bits in the computer, but is not able to accommodate the later development of open and modularised architectures of the web. Reusable information and components, and an increased emphasis on user interface - characteristics of web architectures - are not mapped into the building metaphor of the Zachman framework.

MDA separates models into the CIM, PIM and PSM. Web systems however contain strong elements of creative design which is not handled by the MDA models. MDA is based on use of the Unified Modeling Language (UML) which has become the industry standard notation. But UML is insufficient to model aspects such as the user interfaces and some business aspects of Web systems. UML and MDA are not fully understood by the IT community (Ambler 2004). UML has generally been found lacking as a basis for overall IT architecture design (Spencer 2000).

We propose a Web Application Architecture Framework to fill in the identified gaps in the existing Architecture Frameworks. The unique technical characteristics of web system (Lowe et al. 2001) are fused into this framework.

3. Web Application Architecture Framework (WAAF)

The Web Application Architecture Framework proposed here classifies concerns related to the development of web systems along two dimensions. As shown in Table 1, the horizontal dimension (rows) concerns the perspectives of the different participants in the web application development process. The perspectives are those of: business owners; web system users; information architects; system architects; developers; and testers. The vertical dimension (columns) classifies the architectures into four categories, namely: structure (what); behaviour (how); location (where); and pattern. The first three categories (What, How, Where) mirror those in the Zachman framework. The fourth category (Patterns) has been added based on the growing recognition of the the importance of patterns in software systems generally, and Web systems in particular (Platt 2002, Montero, 2003). Each cell in the framework is a model, a description, or an architecture as appropriate.

The classification in columns of the WAAF Matrix is described in the following abstractions:

Structure: The abstraction of the entities comprising the system and the inter-relationships between these entities.

Behaviour: The description of the functional workflow processes of the system. The “*Behaviour*” column specifies the nature of the interactions amongst the entities that are described in the *Structure* column.

Location: The description of the physical or logical location of the system entities relative to others. It builds the sense of neighbourhood awareness.

Pattern: This refers to the reuse of real-world experience harvested from best practices for successful, rapid and cost-effective system development (Platt 2002). As existing patterns may not be classified into “structure, behaviour and location”, this column will list and describe patterns in their original way.

Table 1. WAAF Matrix - Web Application Architecture Framework

	Structure (What)	Behaviour (How)	Location (Where)	Pattern
Planning Architecture (Planner's Perspective)	List of things important to the business	List of processes the business performs	List of locations in which the business operates	Possible business models and patterns
Business Architecture (Business Owner's Perspective)	e.g. Business Entity Relationship Model	e.g. Business Process Model	e.g. Business Entity Location Model	e.g. Business Model Patterns
User Interface Architecture (User's Perspective)	e.g. User Interface Structure Model	e.g. User Interface Flow Model	e.g. User Site Map Model	e.g. Interface Templates, Navigation Patterns
Information Architecture (Information Architect's Perspective)	e.g. Information Dictionary	e.g. Information Flow Model	e.g. Information Node Location Model	e.g. Information Scheme Patterns
System Architecture (System Architect's Perspective)	e.g. System Functioning Module/Sub-Module/ Server Page Structure	e.g. Workflow Model of Module/Sub-Module/ Server Page	e.g. Site Mapping Model of Modules /Sub-Modules/ Server Pages	e.g. Design Patterns, Presentation styles
Web Object Architecture (Developers' Perspective)	e.g. Physical Object Relationship	e.g. Algorithms in Source Code	e.g. Network Deployment Model	e.g. COTS, Components, Code Library
Test Architecture (Tester's Perspective)	e.g. Test Configuration	e.g. Test Procedure	e.g. Test Deployment	e.g. Templates, Standards of Test Document

In the following subsections we will present the framework – describing each of the rows (perspectives). The framework is discussed using examples of a commercial web application; an Australian company that specializes in matching investors to entrepreneurs who are seeking investment capital. For confidentiality reasons, we use a fictitious name for the company - “XYZ-Match”.

3.1 Planning Architecture (PA), Planner's Perspective

This perspective is concerned with issues important to planning of the web system.

Cell (PA-Structure) lists the entities important to the business. Business entities may be a person, a thing or a concept that is part of, or interacts with, the business (Proforma 2003). In XYZ-Match, example business entities might include the following:

- Investors
- Entrepreneurs
- XYZ-Match web system

Cell (PA-Behaviour) lists the overarching business processes in which the business participates. In the example of XYZ-Match, "Investor listing information to Venture Capital Directory" is an example of one such business process.

Cell (PA-Location) lists the physical (e.g. specific cities) or logical (e.g. via the internet) locations in which the business operates.

3.2 Business Architecture (BA), Business Owner's Perspective

This perspective models the business structures, processes and locations, and patterns of the web application system from the viewpoint of business owners. Not all parts of the business architectures in an organisation will be transferred into web system architectures. Only relevant structures, behaviours and locations in the Business Architecture (i.e. those parts which relate to the web system) will be considered in this framework.

In some senses this perspective also forms the basis of the understanding of system scope and ultimately enabled a clearer understanding of system requirements from the business perspective.

Cell (BA-Structure) describes the business structure including business entities and their relationships. An example model within this cell could be a business entity-relationship diagram (ERD) that models the business concepts, entities and business rules. In such a model, the business rules capture the relationships between the business entities. Figure 1 is an example of a business ERD of *XYZ-Match*.

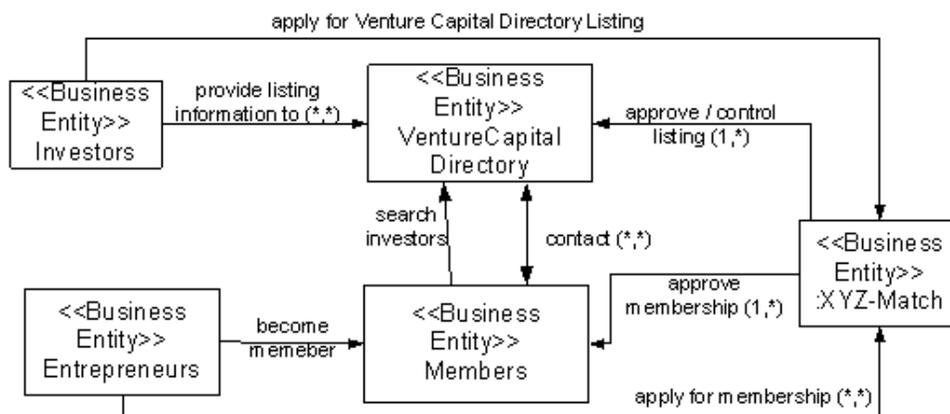


Figure 1. Business structure model

Cell (BA-Behaviour) models the business workflows of the business entities interacting with the business. Flowcharts, activity diagrams, and collaboration diagrams are common tools for business process modelling. If object-oriented technology is applied, an example of a business process model could be a UML use case diagram coupled with sequence diagrams or activity diagrams that describe each business use case. Figure 2 is an example of an activity diagram to model the business use case “Entrepreneurs search and contact investors in Venture Capital Directory” of *XYZ-Match*.

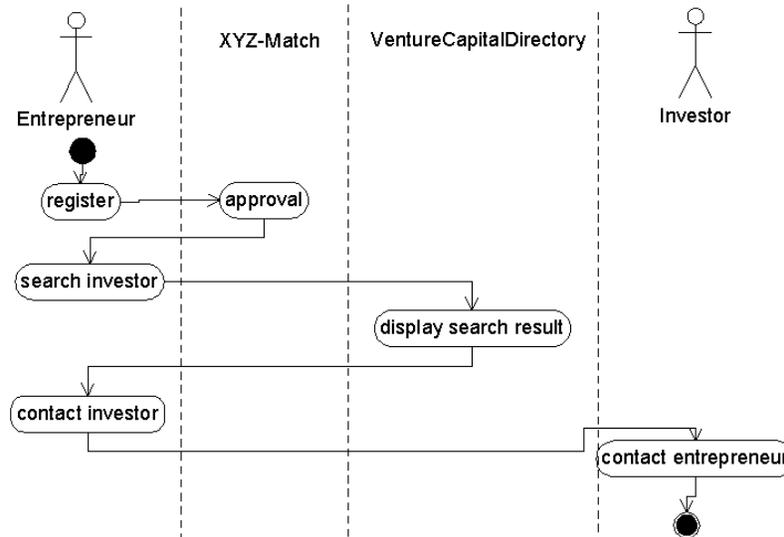


Figure 2. Business process model

Cell (BA-Location) models the locations of business entities. Figure 3 shows the business location model of the entrepreneurs of *XYZ-Match*.

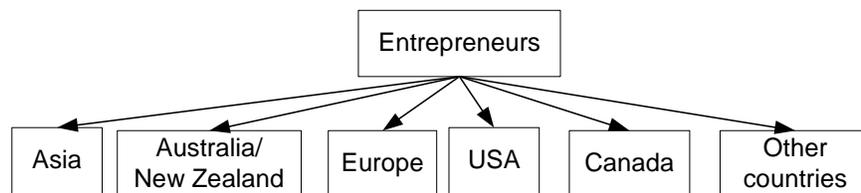


Figure 3. Business entity location model

Cell (BA-Pattern) describes business patterns for the web system. Patterns can be described as “a three-part rule, which expresses a relation between a certain context, a problem, and a solution” (Alexander 1979). Business patterns generalize solutions to solve problems that are common to different business situations. They can be reused repeatedly and can be combined and adapted in many different ways (Eriksson 2000). Example business patterns which might be considered as potentially relevant for *XYZ-Match* include: a brokerage model; an advertising model; a merchant model; an affiliate model; and an application service provider model.

3.3 User Interface Architecture (UIA), User’s Perspective

This row describes the components of the system, their roles and relationships as they are perceived by users of the system.

Cell (UIA-Structure) describes the structure and contents of user interfaces such as HTML pages, user received emails and reports that users will see. There are two levels of user interface structure: page level and site level. At page level, the layout and the contents of each page is defined. Example models at page level include user interface prototypes and web page fragments. At the site level, the composition of entire user experience with the web application and their relationship are presented. Web page class diagrams, user interface prototypes and

presentation views for entire the user experience using IBM OVID (Robert et al. 1998) are examples of user interface structure at site level.

Table 2 is an example of fragments of a web page “VC Firm Listing in VCDirectory” of *XYZ-Match* at the page level.

Table 2. Web page fragment

Logo	Page Title Venture Capital Firms		Advertisement (Business Plan Templates)
Navigation	Introduction Of VCFirms		
	VC Firm Listing		
	Name of VC Firm	Brief Description	
	<<Prev 10 Page scroller Next 10>>		
Disclaimer			
Footer			

Figure 4 is an example of a possible user interface structure of *XYZ-Match* at the level of the entire user experience.



Figure 4. Web page structure for *XYZ-Match* site

Cell(UIA-Behaviour) models how external and internal users access and utilise the web application or system. An overview of user behavioural paths is presented in a user interface flow diagram. Tools like UML state chart diagrams, UML activity diagrams, flowcharts, white site prototypes, user stories, and storyboards can be used here to describe user interaction behaviour.

In many cases, functional prototypes are utilised rather than using formal diagrams. These prototypes might be partial implementations (such as white site prototypes) or artificial examples (such as storyboards or paper prototypes) but both are used to present the paths of user behaviours. Figure 5 is an example of a User Interface Flow Diagram in the VCDirectory module of *XYZ-Match*.

Cell (UIA-Location) describes the location of web pages in the users’ view. A site map is an example which organizes pages taxonomically without showing the details of each page. Figure 6 is part of the site map of *XYZ-Match* web site.

Cell (UIA-Pattern) collects user interface patterns. The SAP INFO glossary (SAP 2004) describes User Interface Patterns as “proven software components that can be used for recurring tasks on the part of the user. In line with the Pattern concept, a User Interface Pattern is defined at various levels on a non-technical basis, and then programmed as a cross-application Pattern”. For web applications, example UI patterns could include UI presentation templates reused from other projects or other modules and navigation patterns. Example web user navigation patterns used in *XYZ-Match* include the directory pattern and the guided tour pattern.

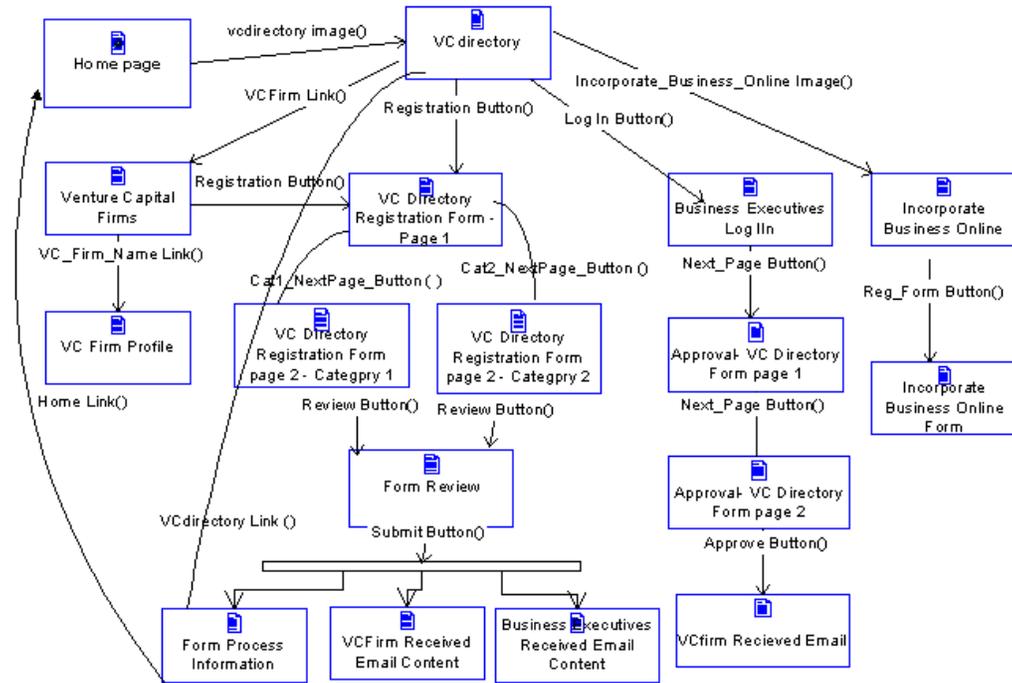


Figure 5. User interface flow diagram

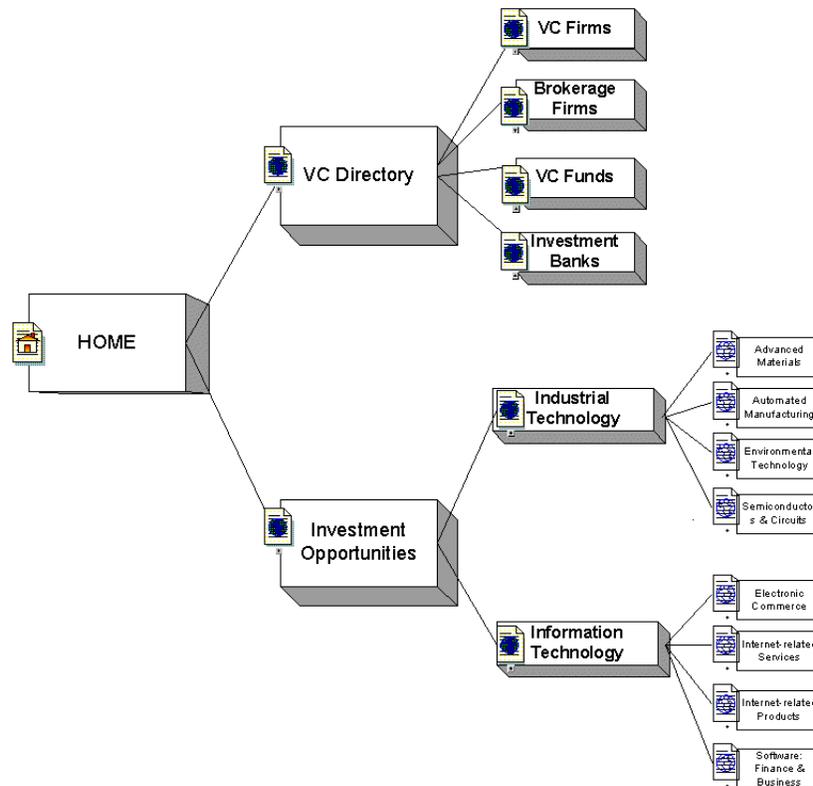


Figure 6. Site map of XYZ-Match

3.4 Information Architecture (IA), Information Architect's Perspective

An information architecture is “the result of the integrated approach to information design”. It is the “blueprint for maximizing software usability via the integrated design of labels, messages, online support elements, and printed support elements” (Henry 1998).

An information node is a representation of an element of architecture to create, process or consume information. Information nodes includes information source and information destination. An information node could be a system, an organization or an external entity.

Information architects can be analogous to the librarians of web development. The concerns of the information architects are to classify and construct the structure, relationship, flow and location of information that is needed in the User Interface perspective to connect the external and internal users to access and operate the content and the functionality of the web application. This perspective is independent from the implementation of the system.

It is also worth noting that a major concern in website maintenance is that of maintaining the information structures. The information architecture captures relevant information and – if modelled appropriately – facilitate important tasks such as effective hypermedia authoring and link maintenance.

Cell (IA-Structure) structures, organizes and labels the information and its interrelationships. Information is “the interpretation of data within a context set by a priori knowledge and the

current environment” (Lowe et al. 1999). A Web information dictionary is an example in this cell. Information could be organized alphabetically, chronologically, geographically, or by topics, tasks, users, metaphor or by hybrid categories. Examples of information labels include contextual links, headings, navigation label, index terms and iconic labels (Rosenfeld et al. 2002). Figure 7 is an example of the information structure of “Information of Venture Capital Directory”.

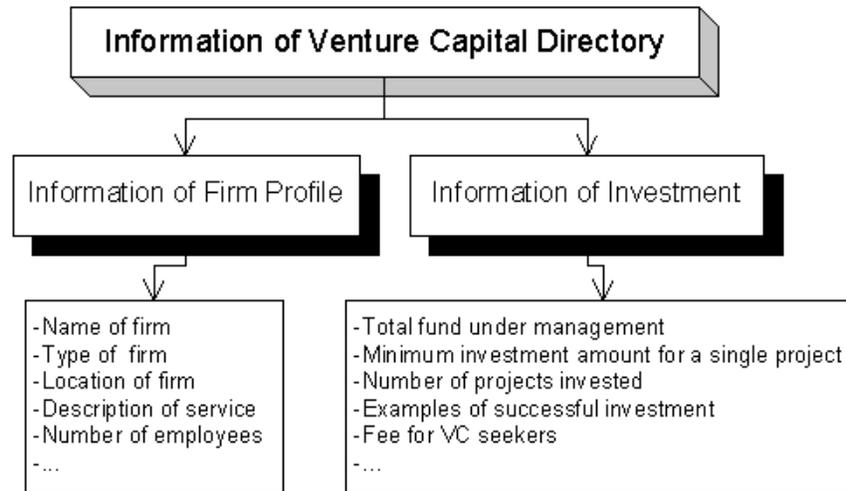


Figure 7. Example of information structure and labels

Cell (IA-behaviour) models the information creation, exchange, process and consumption flow between the system, the organization and external entities, and the triggering events. This information flow is derived from the user interface flow modelled in the User Interface perspective. Example models are an Information Exchange Matrix (Chief Information Officer Council 2000) and an information flow diagram such as WebML+ (Lowe et al. 2003a). We use an information flow diagram with a modified data flow diagram and activity diagram as an example in Figure 8 . In this diagram, the information flow is partitioned to swim lanes by information nodes. Each lane holds the information process activities, information and information repository that belong to that information node. The information flow starts from a start point node and ends with an end point node. The labels of the exchanged information are defined in Cell(IA-structure).

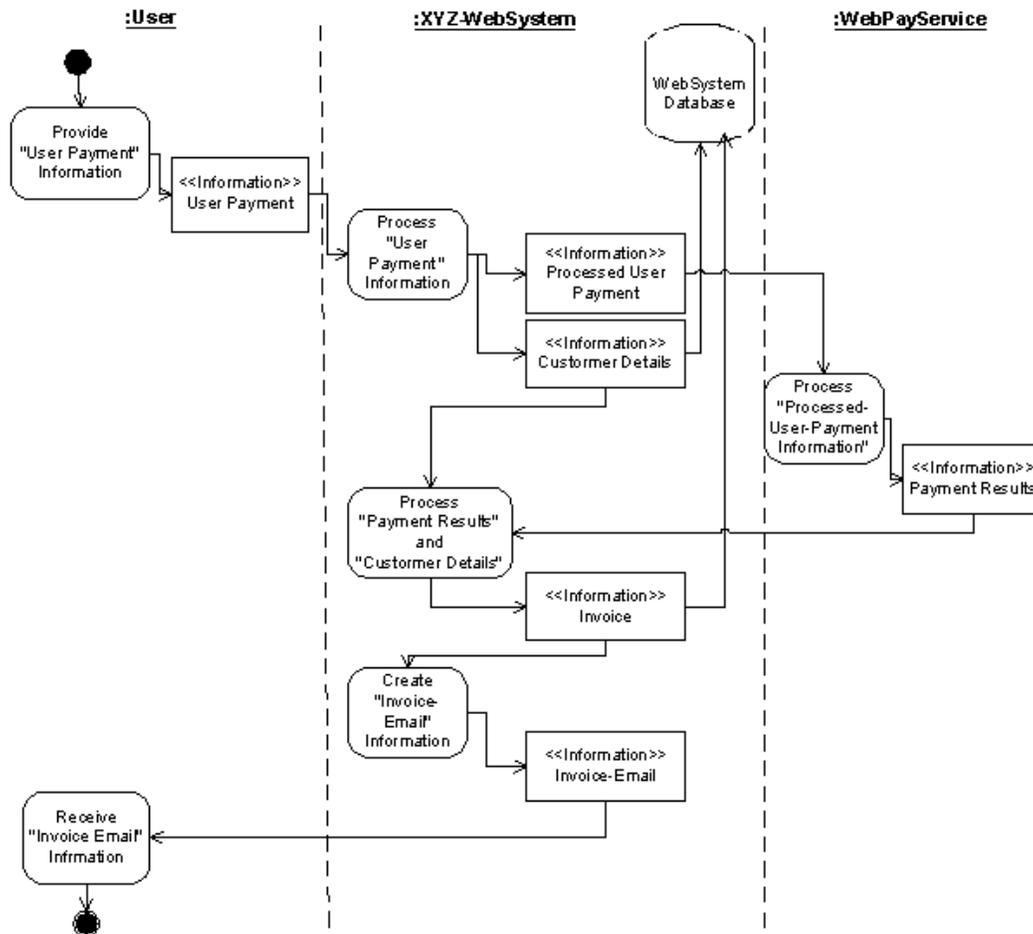


Figure 8. Information flow diagram

Cell (IA-Location) Information could be stored in Information Nodes such as a database, XML files, file folders, or some other repository of external information entities. An Information Node represents both information source and destination. In this cell, the location and the relationship of the Information Nodes are modelled. As this perspective is free from the implementation of detailed information repository, the information location model is only at the context level. Figure 9 uses a deployment diagram to describe a location model.

Cell (IA-Pattern) describes patterns of constructing information structure and information flow. For example, information structure pattern can be constructed from a layered classification scheme for key web characteristics (Lowe et al. 2001).

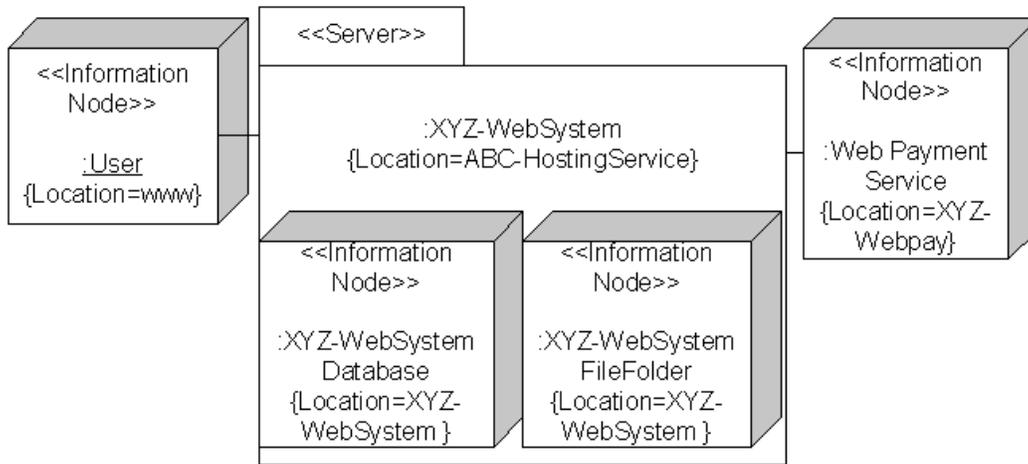


Figure 9. Location model of information source and destination

3.5 System Architecture (SA), System Architect's Perspective

System architects focus on the overall architecture of the system as a basis for integrating various system perspectives. For example, a web system could be designed into layers. Major functioning modules are grouped at the top layer. Each module is decomposed into a number of sub-modules at a middle layer. Sub-modules comprise a set of server pages or server files on a lower layer to fulfill the functionalities.

Cell (SA-Structure) specifies the structure, the responsibilities and the relationships of the design elements of a web system. Take *XYZ-Match* as an example; there are three layers: module layer; sub-module layer; and server page layer. Within each sub-module, the structure of a set of low-level components to fulfill the functionalities of each sub-module is described. For each component (such as a server page) we might typically specify inputs (either from other components or from the user-interface), the responsibilities, and the outputs (to the user interface, or to other components).

An example of the server page structure of “Sub-Module 1.1: VC Directory Listing” can be seen in the bottom layer of Figure 10. Five major server pages are listed in this structure. In this example, the responsibilities of each server page are described on the top of the server page as comment lines. Source code for server pages should not be included in this perspective. If an object orientated design is adopted, class diagrams and package diagrams of modules, sub-modules and server pages can be used to specify the structure in this cell.

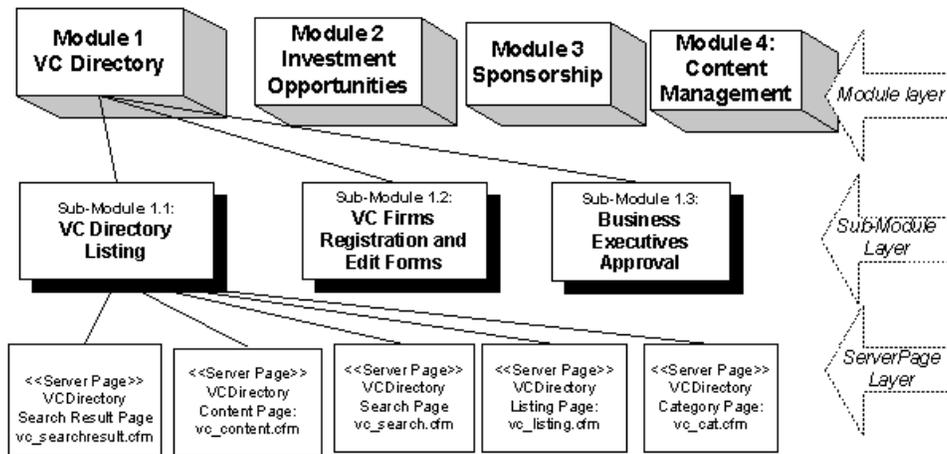


Figure 10. Three layers of system architecture of *XYZ-Match*

Cell (SA-Behaviour) specifies system workflows which implement business logic within the modules and sub-modules. Figure 11 demonstrates the workflow within sub-module “VC Directory” using a server page flow diagram.

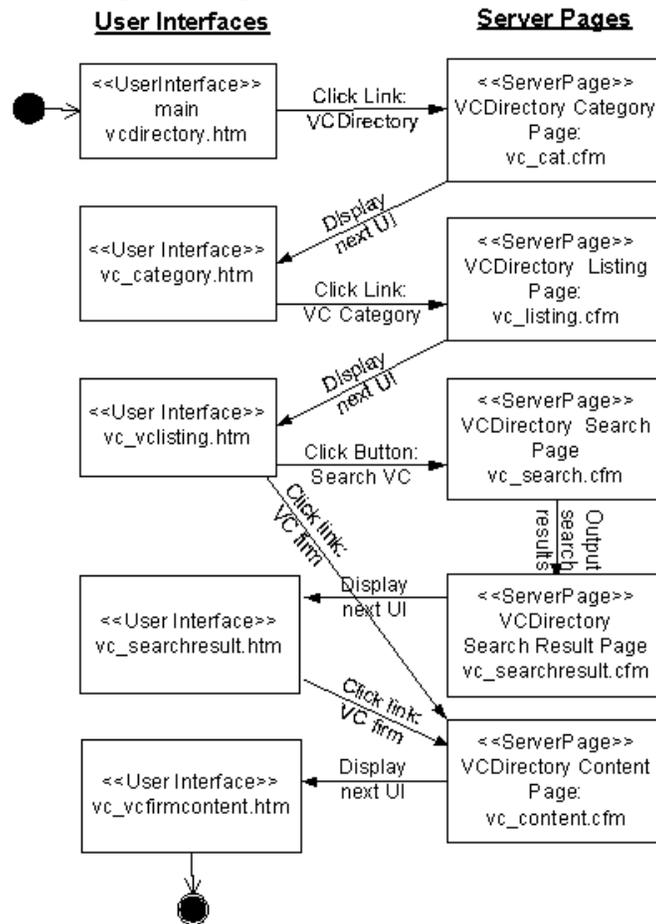


Figure 11. Server page flow diagram

Cell (SA-Location) maps design elements such as modules, sub-modules and components (e.g. server pages) to their physical locations. For example, modules may be located in parallel directories. Sub-modules can be under the sub-directory of their parent modules. Server pages or server files may be under their sub-module directories. Reusable server pages can be grouped into a directory. For large web applications, if the rule of separation of business logic and presentation is applied, server pages to deal with business logic, user interface views, and customized workflow processes can be located in different directories. Figure 12 is an example of directory mapping.

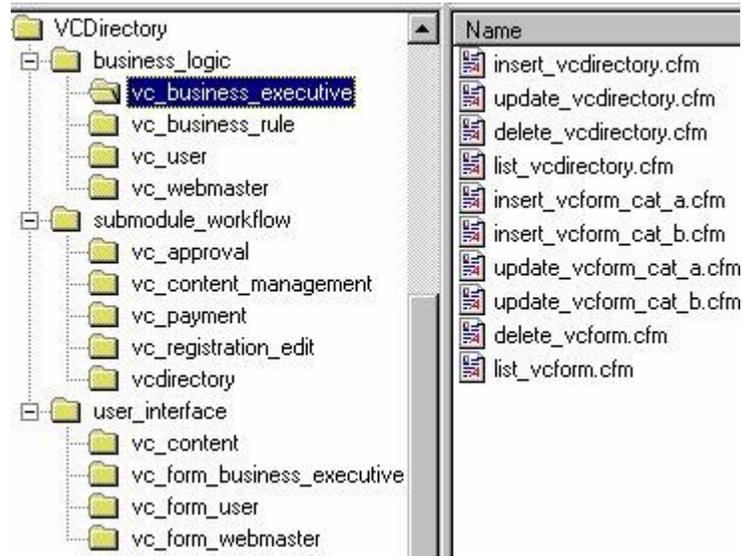


Figure 12. Directory mapping

Cell(SA-Pattern) lists system design patterns and presentation style patterns. J2EE Blueprints (Sun Microsystems Inc. 2001), Apache Struts (Apache Software Foundation 2003) and Coldfusion Fusebox (Peters et al. 2002) provide example design patterns. For presentation patterns, styles could be defined for each module or sub-module style sheets or templates. The architectures of such style sheets or templates can be described in this cell.

3.6 Web Object Architecture (WOA), Developer's Perspective

The system design is implemented via source code and other web objects. This row describes the architecture of these implementation artefacts from the developer's perspective. Examples of web objects include ActiveX components, COTS (commercial-off-the-shelf) components, objects like shopping carts, multimedia files such as video, plug-ins, data tables, server page files (source code), Applets, agents, guards, graphics files, and scripts, etc. (Reifer 2000).

Cell (WOA-Structure) defines input/output data or information for each implementation object. Web objects relationships used in the source code are also specified. Examples of such input/output data definitions are on the top of each server page.

Web object dependency graphs are an example of how the relationship of web objects can be captured. A change to one web object will require appropriate change in its dependent objects identified in the graph. Figure 13 is part of a web object dependent graph of *XYZ-Match*.

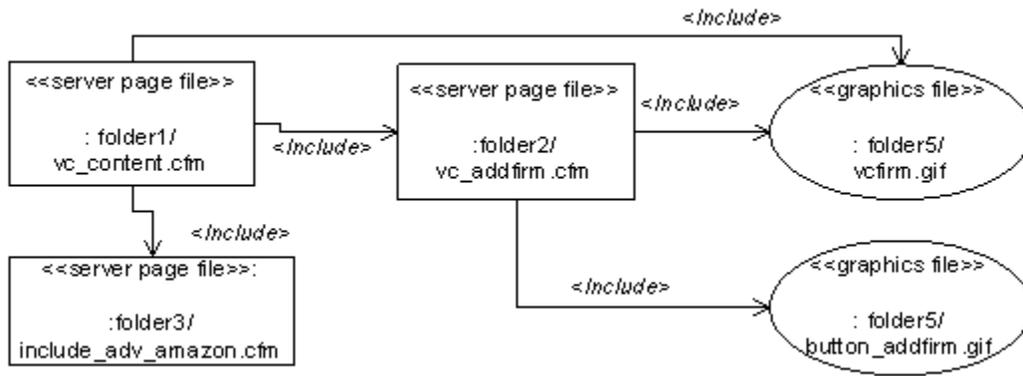


Figure 13. Web object dependency graph

Cell (WOA-Behaviour) describes the algorithms within the source code. Detailed algorithms of code flow are typically self-documented within the source code.

Cell (WOA-Location) physically allocates the web objects in the web network. A deployment diagram is an example tool to model network location.

Cell (WOA-Pattern) Web systems are commonly built using open source code and components. Source code could be reusable from one project to other projects and from one programmer to other programmers. This cell utilizes the reusability of programmer's work. Examples are COTS components, internal components, open source code, code libraries, custom tags and code snippets.

3.7 Test Architecture (TA), Tester's Perspective

Testing web applications includes verification and validation of artefacts produced in the rows above this perspective, and seeks to identify possible weak points in the system design. Testers look at the web product in terms of both the web project participants' perspective and possible hackers' perspective – i.e. valid system users and invalid system users.

Tests of the business architecture, user interface architecture, information architecture, and system design architecture will most commonly be static tests (such as a review or walkthrough). Tests of the web object architecture are more likely to be dynamic tests by execution of the web product (as some combination of unit tests, integration tests, load tests or stress tests, system tests which include alpha testing in the developer's environment and beta testing in the client's environment).

The Tester's perspective in this row includes the structure of test documents, test procedure, location model and patterns in different types of tests. A test can be a black-box test from a users' view or a white-box test from viewpoints of internal participants or external hackers.

Cell (TA-Structure) defines and organizes test data and test documents. Example test documents include a test plan, the items under test, test designs, test cases, test data, test procedures, test logs, and test reporting (such as test item transmittal reports, test incident reports and test summary report) (IEEE 1999). The relationships between the test documents and items can be described in a test configuration file or diagram. Test item interfaces to other web objects, sub-modules, modules or stubs are also defined here. An example of a test document relationship

diagram is shown in Figure 14. A Test Harness Graph can be used to define the interfaces and relations of test system.

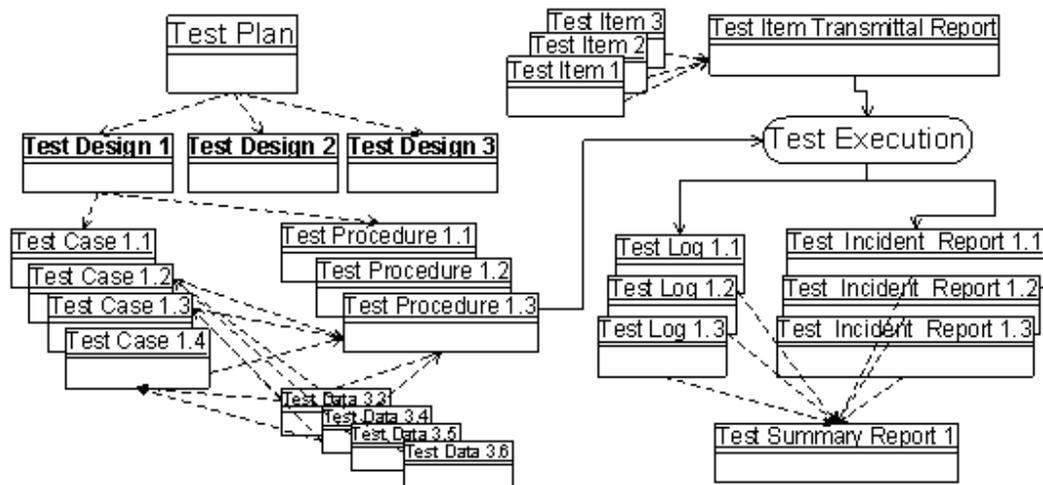


Figure 14. Test documents relationship diagram

Cell (TA-Behaviour) models the test execution process using test procedures that configure a set of test data and test cases. Test Flow Graphs and UML interaction diagrams are common tools to model the flow of test cases and test data in each test execution. User Interface Flow Diagrams (defined in the UI architecture) could be used as a Test Flow Graph in a black-box test to validate the behaviour of the user interface flow. In each test case we would typically specify: test sequence; alternatives; loops and defaults of stimuli to and observations from the test items. Test step flow inside each test case could be based on the algorithm in a server page file for white-box testing.

Cell (TA-Location) maps the test execution to network components. To test web products, the test configuration for a test described in test structure is deployed into the network. The deployment of the test execution of a test on certain nodes in a test environment (such as on the development server or in a network of client's production servers) is described in this cell.

Cell (TA-Pattern) describes test patterns. Some of the test documents could be reused across projects and organizations. For example, test documents for a particular test can be tailored from standards or templates. Test cases and test data can be reused by other test designs. Other aspects potentially included in this cell include templates or standards of test documents, reusable test data, test cases, and test configurations.

4. Discussion

4.1 Key Attributes Of The Proposed Framework

Each column has a unique model. Each row presents a unique perspective.

The classifications of structure, behaviour, location and pattern are unique and should be independent from other columns in the WAAF Matrix. Similarly, each participant looks at the system from a unique viewpoint. The web application system architecture is represented by the integration of the rows and the columns. The framework reduces the system complexity by

decomposition of the system architectures to level of cells that are orthogonally allocated by column and row.

The framework does not imply the order of the perspective.

The order of the perspectives presented above does not imply a sequence for the development process. Nor does it define a development process. Developers could apply a traditional waterfall development, or equally they could adopt an iterative process to fit their projects. For small and medium web projects, some experienced developers could jump directly to the Web Object Architecture, ignoring other perspectives as they may use mental architectures and document other perspectives later (Kong et al. 2004). Each perspective could be applied to both development of new web systems and maintenance of existing systems.

Allowing existing development paradigms.

This framework is not affixed to any of the existing development paradigms (e.g. object-oriented vs structured design) and thus allows the adoption of any paradigm. Illustration of examples used in each cell does not imply using one method in the entire framework. For example, in *Cell(structure)* of each perspective, if an object-oriented paradigm is used, a sample architecture could be modelled by UML class diagrams. If another paradigm is adopted, models and notations of the paradigm could be used.

It is not necessary to provide models or documentation for all cells.

This framework classifies the architectural constructs according to different perspectives. It does not require heavyweight documentation. Not all cells, columns or rows are needed for a specific web project. Organizations can tailor the framework to fit the needs of their systems. For example, the information architecture and the system architecture in this framework could be merged into one perspective (Lowe et al. 2003c). For small and medium web applications, experienced developers could work with business owners to merge cells in this framework to fit the project need. Work on a Critical Feature Matrix (Kong et al. 2004) is an example where all cells of this framework have been integrated into a lightweight matrix.

4.2 Additional Comments

It is also worth noting that we have focused on the particular design perspectives which exist with Web systems. We have not considered the *process* by which these designs or architectures are constructed. We believe that they are just as applicable whether a conventional waterfall approach is adopted, or a more contemporary agile methodology.

Similarly, approaches such as participatory design or user centred design will still have outcomes which map into the perspectives and layers of abstraction which we have defined. Different approaches will, however, place a stronger emphasis on different perspectives. For example, user-centred design would be likely to emphasise the *User Interface Architecture (User's Perspective)*. Our framework however support understanding with regard to the related perspectives and how they might connect to those which are preferred.

5. Conclusion and Future Directions

Web systems have characteristics which distinguish them from other systems. Web systems typically face high levels of client uncertainty of their needs and also in understanding whether a design will satisfy their needs. They have high levels of requirements volatility and project scope change due to the evolution of business models. They have shorter delivery times, and demand fine-grained evolution and maintenance with an ongoing process of content updating, editorial changes and interface tuning (Lowe et al. 2001).

This paper presents an Architecture Framework for web application which is based on the separation of concerns and takes into account these unique characteristics of web systems. The framework has two dimensions in a matrix structure. One dimension concerns domain of consideration (structure, behaviour, location, and pattern) of the web system. The other orthogonal dimension concerns the perspectives of various participants of the system. This framework can serve as a strategic guide to the development of web systems. It can also be used as a tool for analysis and re-engineering of existing web systems.

Future research will focus on modelling the organizational characteristics of web application systems into “why (motivation model), who (role model), when (scheduling model), how much (cost model)” for each perspective. The framework in this paper assumes the target is the web application. Future research direction could be establishing an architecture framework for web services in the similar perspectives and classification focus on the unique characteristics of web services.

References

- Alexander, C. (1979) *Timeless way of building*. New York, Oxford University Press.
- Ambler, S. W. (2004) Agile modeling, <http://www.agilemodeling.com/>
- Apache Software Foundation. (2003) Struts, Version 1.1, <http://jakarta.apache.org/struts/>
- Chief Information Officer Council. (2000) TEAF, Treasury Enterprise, Architecture Framework, Version 1.
- Chief Information Officer Council. (2001) FEAF, The federal government enterprise framework.
- Department of Defence. (1997) C4ISR Architecture Framework, Version 2.0.
- Eriksson, H.-E. (2000) *Business modeling with UML: business patterns at work*. New York, John Wiley & Sons.
- Frankel, D., Harmon, P., Mukerji, J., Odell, J., Owne M., Rivitt, P., Rosen, M., and Soley, R. (2003) The Zachman Framework and the OMG's Model Driven Architecture, *Business process Trends*.
- Henry, P. (1998) *User-centered information design for improved software usability*. Boston, Artech House.
- IEEE. (1991) 829-1983 (R1991) IEEE Standard for Software Test Documentation.
- IEEE. (2000) IEEE Recommended practice for architectural description of software-intensive systems. 1471-2000.
- Kong, X., Liu, L. and Lowe, D. (2004) Critical Feature Method - A Lightweight Web Maintenance Methodology for SMEs. *Journal of Digital Information*, Volume 5, Issue 2.
- Kruchten, P. (1995) "The 4+1 view model of architecture." IEEE Software: 42-50.
- Lowe, D., and Hall, W. (1999) *Hypermedia and the Web: An Engineering Approach*. New York, John Willey & Sons Ltd.
- Lowe, D., and Henderson-Sellers, B. (2001) Impacts on the development process of differences between web systems and conventional software systems. *SSGRR 2001: International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet*, L'Aquila, Italy.
- Lowe, D., and Tongrunrojana, R. (2003a) WebML+ in a nutshell: Modelling Architectural-Level Information Flows. *WWW2003: 12th International World Wide Web Conference*, Budapest, Hungary.
- Lowe, D., and Henderson-Sellers, B. (2003b) Characterising Web Systems: Merging Information and Functional Architectures. *Architectural Issues of Web-Enabled Electronic Business*. V. K. S. Murthy, N. Hershey, PA, USA, Idea Group Publishing.
- Montero, S., Díaz, P., and Aedo, I. (2003) Formalization of Web Design Patterns Using Ontologies. *AWIC 2003*: 179-188
- Object Management Group. (2001) Model Driven Architecture, <http://www.omg.org/mda/>

- Peters, J., and Papovich, N. (2002) *Fusebox: developing ColdFusion applications*. Indianapolis, Ind, New Riders.
- Platt, M. (2002) Microsoft Architecture Overview.
- Proforma Corporation. (2003) Enterprise application modelling, <http://www.proformacorp.com/downloads/whitepapers.asp>
- Reifer, D. J. (2000) Web development: estimating quick-to-market software. *IEEE Software* 17(6): 57 - 64.
- Robert, D., Berry, D., Isensee, S., and Mullaly, J. (1998) *Designing for the User with OVID*. New Riders.
- Rosenfeld, L., and Peter, M. (2002) *Information architecture for the World Wide Web*. Cambridge, Mass, O'Reilly.
- SAP. (2004) SAP INFO glossary, <http://www.sap.info/public/en/glossary.php4/list/>
- Soni, D., Nord, R.L. and Hofmeister, C. (1995) Software architecture in industrial applications. *Proceedings of the 17th International Conference on Software Engineering*, Seattle, Washington, USA, ACM Press.
- Sowa, J. F., and Zachman, J. A. (1992) Extending and Formalizing the Framework for Information Systems Architecture. *IBM Systems Journal* 31 (3).
- Spencer, J. (2000) Architecture Description Markup Language (ADML), Creating an Open Market for IT Architecture Tools, <http://www.opengroup.org/architecture/adml/background.htm>
- Sun Microsystems Inc. (2001) J2EE Blueprints. <http://java.sun.com/blueprints/>
- The Open Group. (2003) TOGAF, The Open Group Architecture Framework, Version 8.1.
- Zachman, J. A. 1987. A Framework for Information Systems Architecture. *IBM Systems Journal* 26,(3).