

# A Web Application Architecture Framework

Dr. Xiaoying Kong [[HREF1](#)], Faculty of Engineering, University of Technology, Sydney [[HREF2](#)]  
[xiaoying.kong@uts.edu.au](mailto:xiaoying.kong@uts.edu.au)

Dr. Li Liu [[HREF3](#)], Project Management Graduate Programme, The University of Sydney [[HREF4](#)]  
[l.liu@staff.usyd.edu.au](mailto:l.liu@staff.usyd.edu.au)

## Abstract

As the complexity of web systems grows, there is a need to develop an Architecture Framework to support and guide an organization for web system planning, design, building, deployment and maintenance. This paper presents an Architectural Framework to classify architectures taking into account characteristics of web systems. The framework establishes a two dimension matrix. One dimension separates the concerns of different participants of the web system into perspectives. The other dimension classifies each perspective into "structure (what), behaviour (how), location (where) and pattern". The framework is illustrated by examples from a commercial web application.

## 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 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 needs of the business (The Open Group 2003).

There are various 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) received significant attention. Nevertheless, these Architecture Frameworks were developed for conventional enterprise information systems that do not have major web components. As the complexity of web systems grows, there is a need to develop an Architecture Framework to support and guide an organization for web system planning, design, building, deployment and maintenance.

This paper presents an Architecture Framework for web applications taking into account unique characteristics of web systems. This framework is established in a two dimensional matrix. Each row of the matrix contains a set of architectures by separating the concerns of different participants into perspectives. Each perspective is classified to four categories "structure (what), behaviour (how), location (where) and pattern".

In the following sections, literature review on Architecture Frameworks is conducted in Section 2. Section 3 presents a Web Application Architecture Framework. Section 4 discusses the findings and implications. Finally in Section 5, the conclusions are drawn and future research directions are discussed.

## 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 framework has become the basis for many variations of Architecture Frameworks. An organization does not have a single architecture, but has a whole range of architectures representing different perspectives and different stages. Common in the frameworks, an organization's information system is represented by a range of architectures and stakeholders' perspectives. The Zachman framework presents two dimensions. The first dimension describes perspectives of stakeholders of the system, includes ballpark view, owner's view, designer's view, builder's view, subcontractor's view, 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 concurrent views: logical view, development view, process view, physical view and scenario view. The logical view supports what services 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 on the software development environment. The physical view describes the mapping of the software onto the hardware. The elements in the 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, software configuration staff use it from the development view. Five views are not fully independent. Not all the views are needed in software architectures (Kruchten 1995).

Similarly, Soni et al (1995) classified software architecture for industrial applications into four categories: conceptual architecture, module interconnection architecture, execution architecture and code architecture. Within each category, the structures describe 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 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 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) are developed for government agencies. C4ISR gives comprehensive architectural guidance for all of DoD 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).

The nature of web systems is very different from conventional software systems. At technical level, web systems typically have tighter linkage between business model and 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, Lowe et al. 2003a).

The existing Architecture Frameworks do not distinguish these characters. For example, 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, components, COTS, patterns, more emphasis on user interface, that are characters of web architectures are not mapped into building metaphor. MDA separates models to CIM, PIM and PSM. Web system contains element of art. Code of MDA is automatically generated by models in CASE tools. The art element of web system is ignored. MDA is based on Unified Modeling Language (UML) which has become the industry standard notation. But UML is insufficient to model user interface and some business software. UML and MDA are not fully understood by the IT community (Ambler 2004).

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 "increased emphasis on user interface", "significant importance of information and content", and "more pronounced open modularised architecture" (Lowe et al. 2001) are fused into this framework within user interface architecture, information architecture and patterns. Testing architecture is modelled to consider the web characteristics that web is open to external users world-wide and is more susceptible to security threats.

### Web Application Architecture Framework (WAAF)

The Web Application Architecture Framework proposed here separates concerns of a web system into two dimensions. As shown in Table 1, the horizontal dimension (rows) concerns the different perspectives of the different participants in the web application development process. The perspectives are from the viewpoints 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". 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:

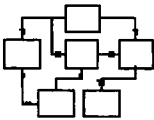
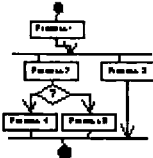
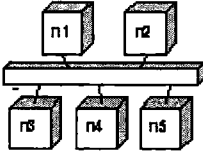

**Structure:** The abstraction of the things comprising the system and their inter-relationship.

**Behaviour:** The description of the functioning workflow process of the system.

**Location:** The location map of the things of the system relative to others geometrically.

**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	-
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 摺 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 摺 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 摺 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 Module/ Sub-Module/ Server Page	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 摺 Perspective)	e.g. Test Configuration	e.g. Test Procedure	e.g. Test Deployment	e.g. Templates, Standards of Test Document

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

### Planning Architecture (PA), Planner's Perspective

This perspective concerns with issues important to planning of the web system. Pattern is not concerned in this perspective.

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

- Investors
- Entrepreneurs
- "XYZ-Match" web system

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

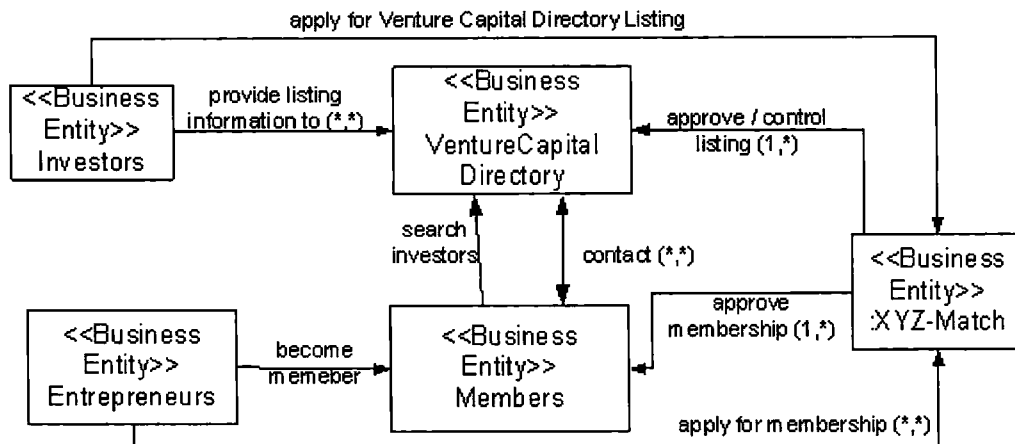
**Cell (PA-Location)** lists the locations where the business operates. E.g. the business branches of XYZ-Match in various locations of the world.

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

This perspective models the business structure, process and locations, patterns of the web application system from the viewpoint of business owners.

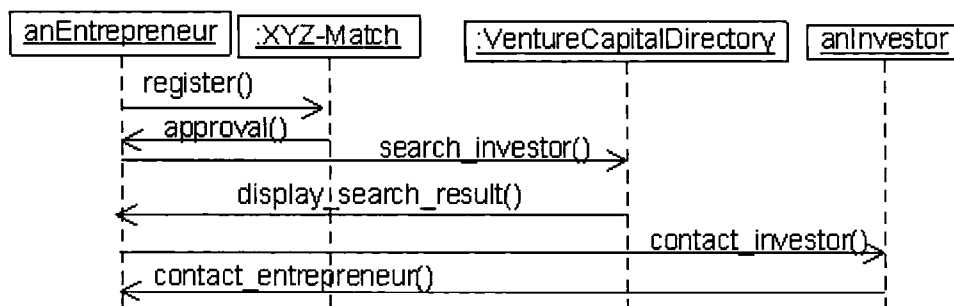
**Cell (BA-Structure)** describes the business structure including business entities and their relationship. Example model of this cell could be a business entity-relationship diagram (ERD) that

models the business concepts, entities and business rules. In this diagram, the business rule represents the relationship between the business entities. Figure 1 is an example of a business ERD of XYZ-Match.



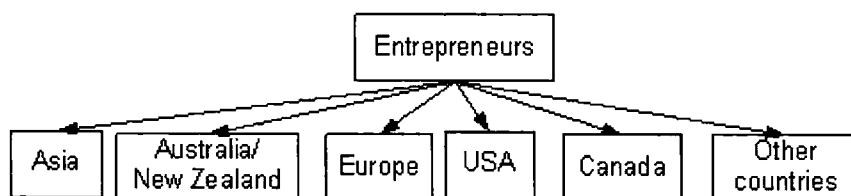
**Figure 1 Business structure model**

**Cell (BA-Behaviour)** models the business workflow of the business entities interacting with the business. Flowchart, activity diagram, collaboration diagram are common tools for business process modelling. If object-oriented technology is applied, example of business process model could be a UML use case diagram and sequence diagrams that realize each business use case. Figure 2 is an example of a sequence diagram to model a business use case "Investor listing information to 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. Pattern 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, et al. 2000). Example business pattern can be an interface metaphor such as the concept of a shopping cart in real shops been used to a shopping

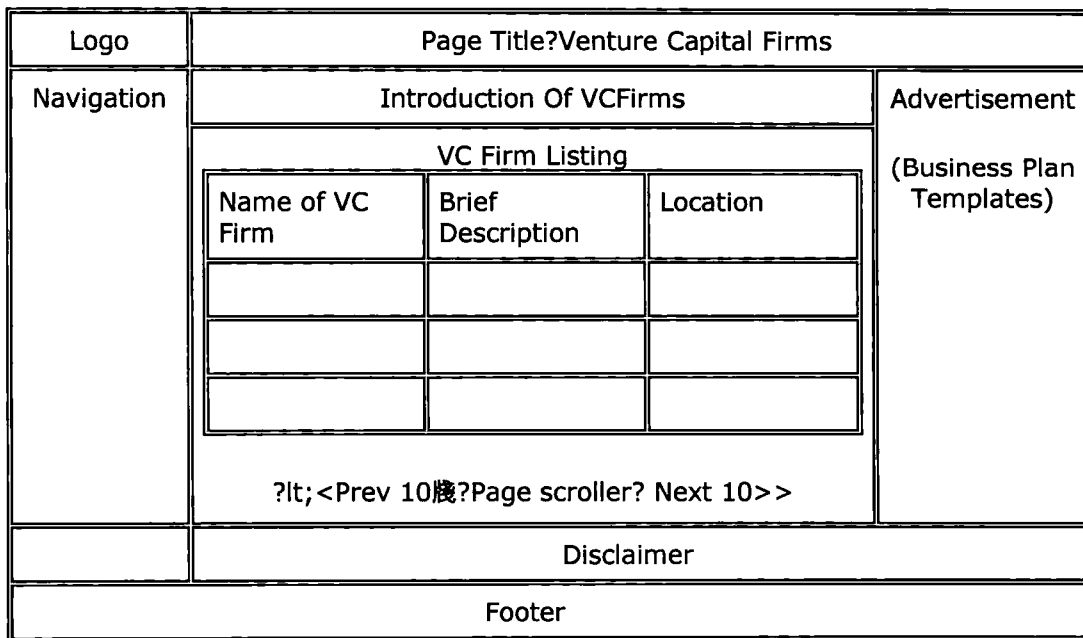
cart solution in an online-shopping web site.

**User Interface Architecture (UIA), User's Perspective**

This row describes the components of the system, their roles and relationships that 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. The composition of web pages and their relationship are defined here. Example models include web page class diagram, user interface prototype, and web page fragments.

Figure 4 is an example of fragments of a web page "VC Firm Listing" in VCDirectory of XYZ-Match.



**Figure 4 Web page fragment**

**Cell(UIA-Behaviour)** models how external and internal users access and operate the web applications system. An overview of user behavioural path is presented in a user interface flow diagram. Tools like UML state chart diagram, UML activity diagram, flowchart, white site prototypes, user stories, and storyboard can be used here to describe user interface behaviour.

In many cases, utilising prototypes rather than using formal diagrams, such as white site prototypes, storyboard, paper prototypes are common in practice to present the paths of user behaviours. Figure 5 is an example of User Interface Flow Diagram in VCDirectory module of XYZ-Match.

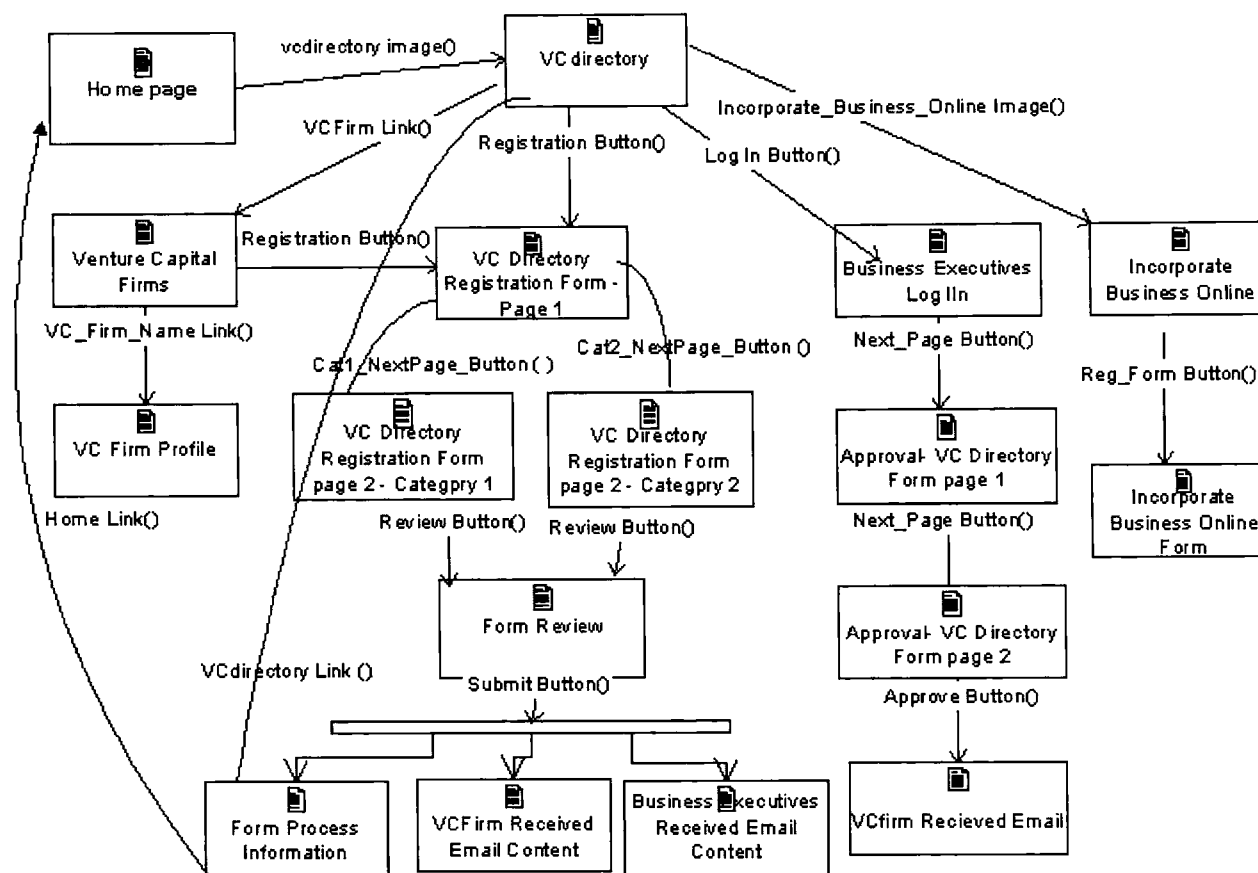


Figure 5 User interface flow diagram

Cell (UIA-Location) describes the location of web pages in users' view. Site map is an example to organize pages taxonomically without showing details of each page. Figure 6 is part of the site map of XYZ-Match web site.

<b>Home</b>									
<b>VC Directory</b>	<ul style="list-style-type: none"> <li>* <a href="#">VC Firms</a></li> <li>* <a href="#">Brokerage Firms</a></li> <li>* <a href="#">VC Funds</a></li> <li>* <a href="#">Investment Banks</a></li> </ul>		Forms <ul style="list-style-type: none"> <li>* <a href="#">Member Registration</a></li> <li>* <a href="#">VC Directory</a></li> <li>* <a href="#">Capital Solicitation</a></li> <li>* <a href="#">Sponsorship</a></li> <li>* <a href="#">Technology Transfer</a></li> </ul>						
	<b>Investment Opportunities</b>	Industrial Technology		<ul style="list-style-type: none"> <li>* <a href="#">Advanced Materials</a></li> <li>* <a href="#">Automated Manufacturing</a></li> <li>* <a href="#">Environmental Technology</a></li> <li>* <a href="#">Semiconductors &amp; Circuits</a></li> </ul>	<table border="1"> <tr> <td>FAQ</td> <td></td> </tr> <tr> <td>Services</td> <td></td> </tr> <tr> <td>Forums</td> <td></td> </tr> </table>	FAQ		Services	
FAQ									
Services									
Forums									
	Information Technology	<ul style="list-style-type: none"> <li>* <a href="#">Electronic Commerce</a></li> <li>* <a href="#">Internet-related Services</a></li> <li>* <a href="#">Internet-related Products</a></li> <li>* <a href="#">Software: Finance &amp; Business</a></li> </ul>							

**Figure 6 Site map of XYZ-Match**

**Cell (UIA-Pattern)** collects user interface patterns. SAP INFO glossary describes User Interface Pattern 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" (SAP 2004). For web applications, example UI patterns could be UI presentation templates reused from other projects or other modules and navigation patterns.

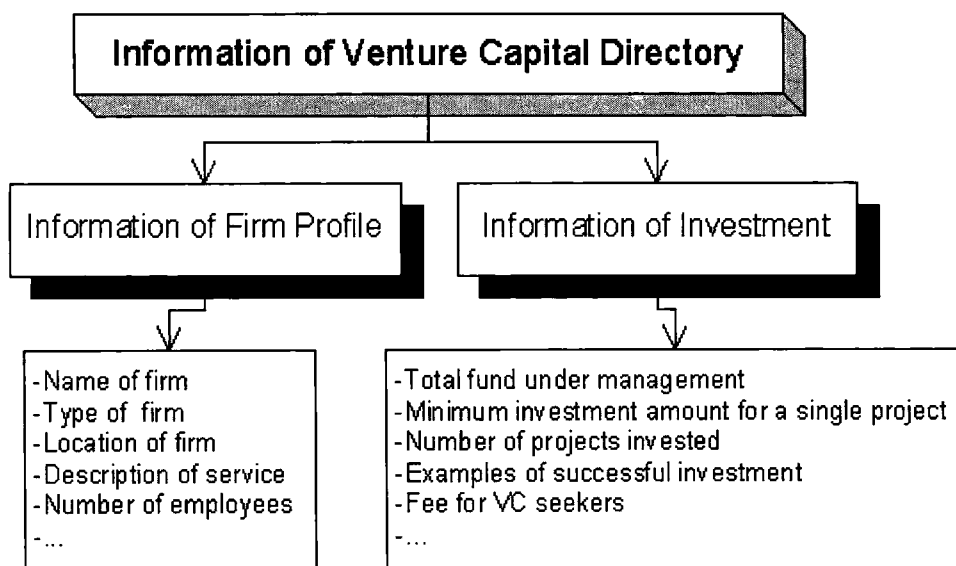
### Information Architecture (IA), Information Architect's Perspective

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).

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

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 are needed in the User Interface perspective to connect the external and internal users to access and operate the contents and the functionality of the web application. This perspective is independent from the implementation of the system.

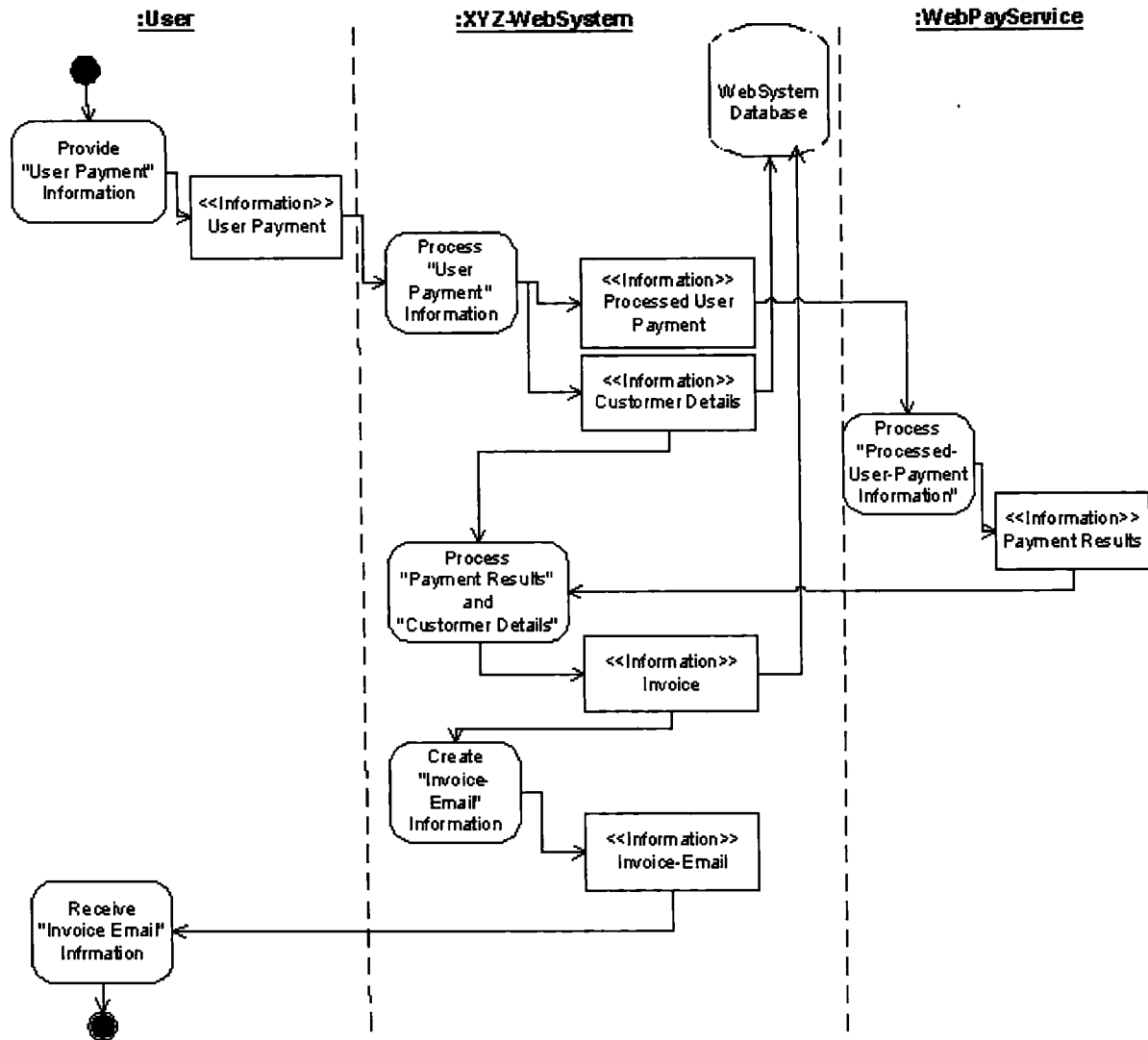
**Cell (IA-Structure)** structures, organizes and labels the information and their relationship. Information is "the interpretation of data within a context set by a priori knowledge and the current environment" (Lowe et al. 1999). Web information dictionary is an example in this cell. Information could 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)** In this cell, the information creation, exchange, process and consumption flow between the system, the organization and external entities, and the triggering events are modelled. This information flow is derived from the user interface flow modelled in the User Interface perspective. Example models are Information Exchange Matrix (Chief Information Officer Council 2000) and information flow diagram such as WebML+ (Lowe et al. 2003b). We use an information flow diagram with a modified data flow diagram and activity diagram as an example in

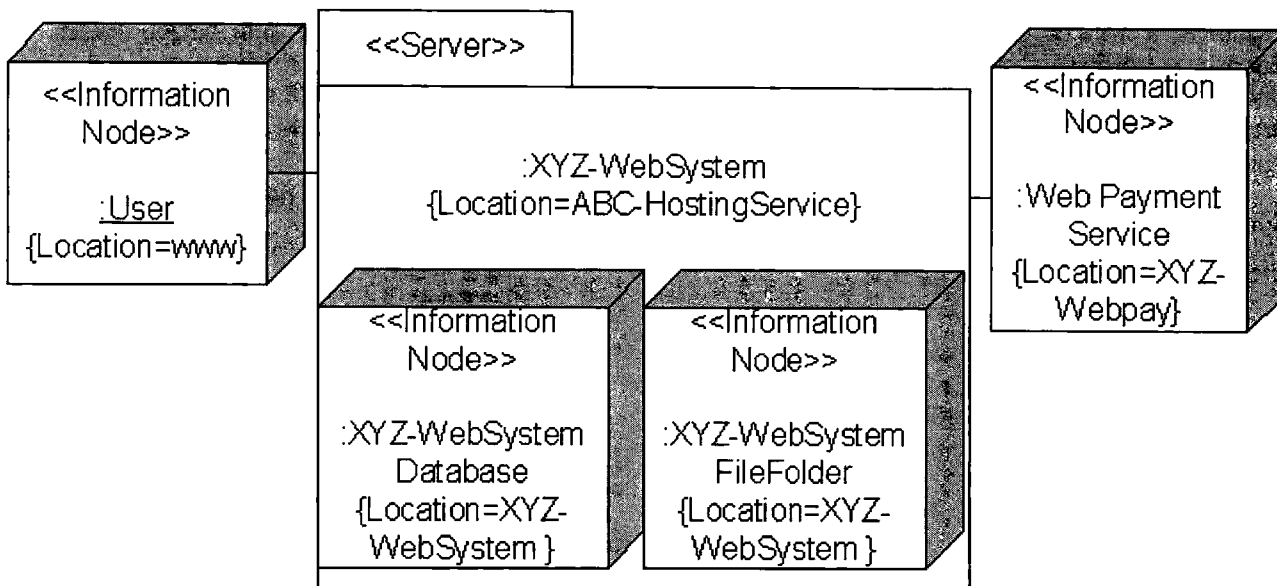


Figure 8. In this diagram, the information flow is partitioned to swim lanes by information nodes. Each lane holds the information process activities, information and information repository that belong to that information node. The information flow starts from a start point node and ends with an end 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 database, XML files, file folders, or other repository of external information entities. 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.



**Figure 9 Location model of information source and destination**

**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).

### **System Architecture (SA), System Architect's Perspective**

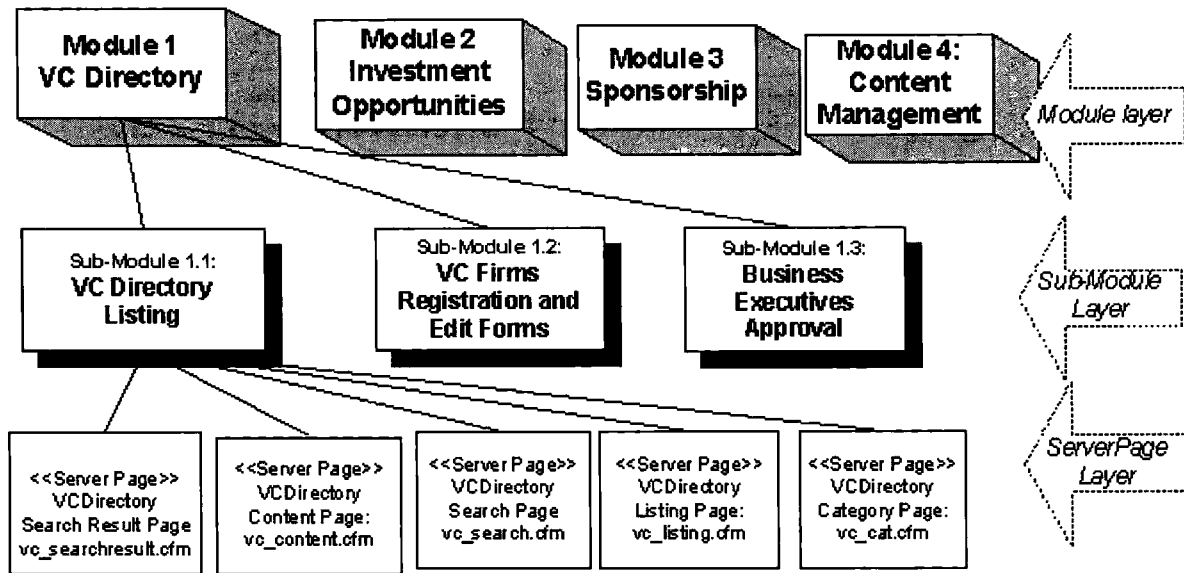
System architects look into the web application system in this perspective for system design. For example, a web system could be designed into layers. Major functioning modules are grouped at top layer. Each module is decomposed into a number of sub-modules at middle layer. Sub-modules comprise a set of server pages or server files on lower layer to fulfil the functionalities.

Architectures of "structure, behaviour, location and pattern" of these design elements are described below.

**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 sub-modules, the input from the pervious user interface, server file or other component, the responsibilities, and the output to the next user interface, server file or component are specified for each serve page.

If object orientated design is adopted, class diagrams of modules, sub-modules and server pages can be used to specify the structure in this cell.

Figure 10 demonstrates the three layer structure of XYZ-Match. The major modules are grouped on the top layer.

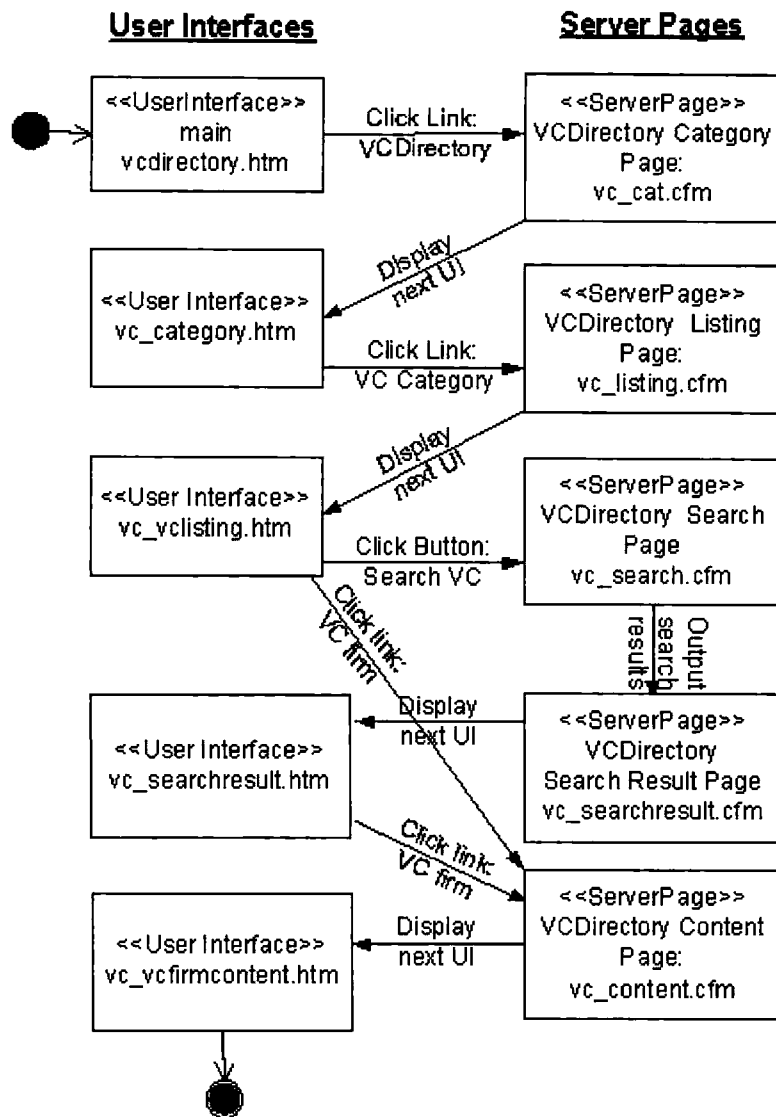


**Figure 10 Three layers of system architecture of XYZ-Match**

Each module is decomposed into a set of sub-modules at middle layer. Figure 10 presents an example of the structure of the sub-module "Sub-Module 1.1: VC Directory Listing" of XYZ-Match.

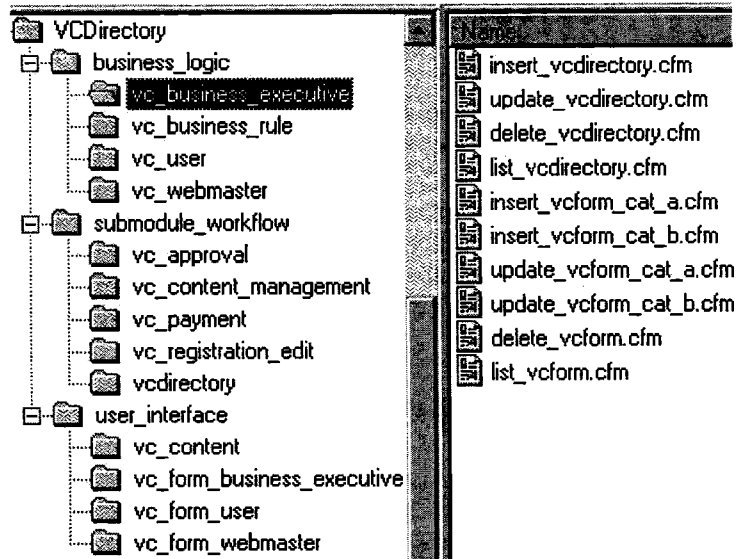
Within each sub-module, the structure of a set of server pages to fulfil the functionalities of each sub-module is described. Example of the server page structure of "Sub-Module 1.1: VC Directory Listing" can be seen in the bottom layer of Figure 10. Five major server pages are listed in this structure. 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.

**Cell (SA-Behaviour)** specifies workflow or 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 module, sub-module and server page to the 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. Reuseable 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 view and customized workflow process can be located in different directories. Figure 12 is an example of directory mapping.



**Figure 12 Directory mapping**

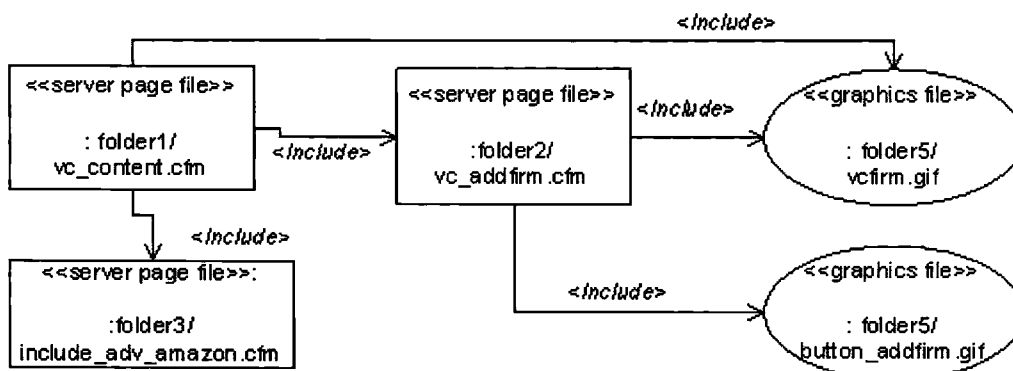
**Cell(SA-Pattern)** lists system design patterns, presentation styles. J2EE Blueprints (Sun Microsystems Inc. 2001), Apache Struts (Apache Software Foundation 2003) and Coldfusion Fusebox (Peters et al. 2002) provide design patterns. For presentation patterns, styles could be defined for each module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

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

System design is implemented into source code and other web objects. This row describes the architecture of source code and other web objects from developer's perspective. Web objects are objects implemented in a web site. Examples of web objects include ActiveX components, COTS 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 source code. Web objects relationships used in the source code are specified. Database tables are web objects. Database scheme is defined and implemented.

Web pages are self-documented. Examples of such input/output data definitions are on the top of each server page. Web object dependency graph is an example to present the relationship of web objects used for the entire web application. The change of one web object will cause the change in its dependent objects shown 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 of source code. Detailed algorithms of code flow are self-documented in source code.

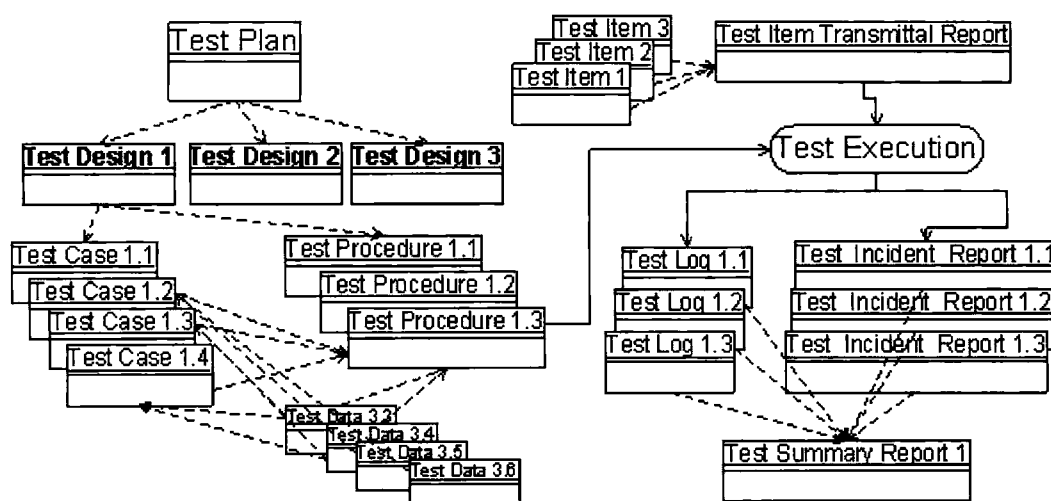
**Cell (WOA-Location)** physically allocates the web objects in the web network. Deployment diagram is an example tool to model network location.

**Cell (WOA-Pattern)** Web systems are commonly built by 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 (commercial-off-the-shelf) components, internal components, open source code, code library, custom tags and code snippets.

## Test Architecture (TA), Tester's Perspective

Web is open to external users world-wide and is more susceptible to security threats. Testing web application includes verification and validation of artefacts produced in the rows above this perspective, and seeks the weak points of the artefacts produced from ROW "Web Object Architecture". Testers look at the web product in both web project participants' perspective and hackers' perspective. Tests of business architecture, user interface architecture, information architecture, system design architecture can be static test such as review or walkthrough. Tests of web object architecture can be dynamic test by execution of web product. Types of such web product tests can be categorized to unit test, integration test which includes module integration test and sub-module integration test, load test or stress test, system test which includes alpha test in developer's environment and beta test in client's environment. 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 documents. Example test documents include test plan, items under test, test design, test case, test data, test procedure, test log, test item transmittal report, test incident report and test summary report (IEEE, 1999). Relationship of the test documents in a test 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. Stubs are dummy items with interfaces only. An example of test documents relationship diagram is shown in Figure 14. 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 configures a set of test data and test cases. Test Flow Graph and UML interaction diagrams are common tools to model the flow of test cases and test data in each test execution. User Interface Flow Diagram defined in UI architecture could be used as a Test Flow Graph in a black-box test to validate the behaviour of user interface flow. In each test case, sequence, alternatives, loops and defaults of

stimuli to and observations from the test items are specified to test steps. Test step flow inside each test case could be similar to algorithm in a server page file.

**Cell (TA-Location)** maps test execution to network. To test web products, the test configuration for a test described in test structure is deployed into network. The deployment of test execution of a test on certain nodes in a test environment such as in development server or in network of client's production server 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. Templates or standards of test documents, reusable test data, test cases, and test configurations can be test patterns described in this cell.

## Discussion

This section discusses 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. For example, the structure model is unique to Column (Structure). It is not repeated in Column (Location). Similarly, each participant looks at the system from a unique viewpoint. For example, system architects deal with "things" as logical design entities. "Things" mean physical web objects for developers.

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. It does not define a development process. Developers could apply iterative process to fit their projects. For small and medium web projects, some experienced developers could jump to Web Object Architecture ignoring other perspectives as they may use mental architectures and document other perspectives later (Kong et al. 2003). 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 entire framework. For example, in Cell(structure) of each perspective, if object-oriented paradigm is used, 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 document 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 web project. Organization can tailor the framework to fit the needs of their systems. For example, information architecture and 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. Critical Feature Matrix (Kong et al. 2003) is an example to integrate cells of this framework into a lightweight matrix.

## Conclusion and Future Direction

This paper presents an Architecture Framework for web application. This framework is based on the separation of concerns and takes into account the unique characteristics of web systems. The framework has two dimensions in a matrix structure. One dimension concerns "structure (what),

behaviour (how), location (where) and pattern" of the web system. Another orthogonal dimension concerns the perspectives of various participants of the system.

This framework can serve as a strategic guide to the development of the web systems. It can also be used as a tool for analysis and re-engineering of existing web systems.

Web systems have other characteristics at organizational level. Such as web systems typically face high level of client uncertainty of their needs and also in understanding whether a design will satisfy their needs; have high levels of requirement, project scope and focus change due to the evolution of business model, have shorter delivery time; demand fine-grained evolution and maintenance with an ongoing process of content updating, editorial changes and interface tuning (Lowe et al. 2001).

Authors are currently working on "why (motivation model), who (role model), when (scheduling model), how much (cost model)" for each perspective to focus on the organizational characteristics of web application systems. This framework assumes the target is the web application at this stage. 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, Available online [[HREF5](#)]

Apache Software Foundation (2003). Struts, Version 1.1, Available online [[HREF6](#)]

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. P. M. (2000). Business modeling with UML : business patterns at work. New York, John Wiley & Sons.

Frankel, D., Harmon, P., Mukerji, J., Odell, J., Owen M., Rivitt, P., Rosen, M., & 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. (2003). Critical Feature Method - A Lightweight Web Maintenance Methodology for SMEs. The Fourth International We-B Conference, Perth, Australia.

Kruchten, P. (1995). "The 4+1 view model of architecture." IEEE Software: 42-50.

Lowe, D., & Hall, W. (1999). Hypermedia and the Web: An Engineering Approach. New York, John Wiley & Sons Ltd.

Lowe, D., & 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., & Eklund, J. (2003a). "Client Needs and the Design Process in Web Projects." Journal of Web Engineering 1(1): ,23-36.

Lowe, D., & Tongrunrojana, R. (2003b). WebML+ in a nutshell: Modelling Architectural-Level Information Flows. WWW2003: 12th International World Wide Web Conference, Budapest, Hungary.

Lowe, D., & Henderson-Sellers, B. (2003c). 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.

Object Management Group (2001). Model Driven Architecture, Available online [[HREF7](#)]

Peters, J. P., Nat (2002). Fusebox: developing ColdFusion applications. Indianapolis, Ind, New Riders.

Platt, M. (2002). Microsoft Architecture Overview.

Proforma Corporation (2003). Enterprise application modelling, Available online [[HREF8](#)]

Reifer, D. J. (2000). "Web development: estimating quick-to-market software." IEEE Software 17 (6): 57 - 64.

Robert, D. B., Dick ; Isensee, Scott ; Mullaly, John; & Roberts, Dave (1998). Designing for the User with OVID, New Riders.

Rosenfeld, L. M., Peter (2002). Information architecture for the World Wide Web. Cambridge, Mass, O'Reilly.

SAP (2004). SAP INFO glossary, Available online [[HREF9](#)]

Soni, D., Nord , R.L. & 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. & Zachman, J. A. (1992). "Extending and Formalizing the Framework for Information Systems Architecture." IBM Systems Journal 31(3).

Sun Microsystems Inc. (2001). J2EE Blueprints. Available online [[HREF10](#)]

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).

## Hypertext References

HREF1

<http://www.eng.uts.edu.au/~xiaoying/>

HREF2

<http://www.uts.edu.au/>

HREF3

[http://www.civil.usyd.edu.au/people/li\\_liu.htm](http://www.civil.usyd.edu.au/people/li_liu.htm)

HREF4

<http://www.usyd.edu.au/>

HREF5

<http://www.agilemodeling.com/>

HREF6

<http://jakarta.apache.org/struts/>

HREF7

<http://www.omg.org/mda/>

HREF8

<http://www.proformacorp.com/downloads/whitepapers.asp>

HREF9

<http://www.sap.info/public/en/glossary.php4/list/>

HREF10

<http://java.sun.com/blueprints/>

## Copyright

Xiaoying Kong, Li Liu, © 2004. The authors assign to Southern Cross University and other educational and non-profit institutions a non-exclusive licence to use this document for personal use and in courses of instruction provided that the article is used in full and this copyright statement is reproduced. The authors also grant a non-exclusive licence to Southern Cross University to publish this document in full on the World Wide Web and on CD-ROM and in printed form with the conference papers and for the document to be published on mirrors on the World Wide Web.