A Web System Trace Model and Its Application to Web Design

Xiaoying Kong, Li Liu* and David Lowe
Faculty of Engineering, University of Technology, Sydney, P.O. Box 123 Broadway Sydney, NSW 2007, Australia
* 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

Traceability analysis is crucial to the development of web-centric systems, particularly those with frequent system changes, fine-grained evolution and maintenance, and high level of requirements uncertainty. A trace model at the level of the web system architecture is presented in this paper to address the specific challenges of developing web-centric systems. The trace model separates the concerns of different stakeholders in the web development life cycle into viewpoints; and classifies each viewpoint into structure and behaviour. Tracing relationships are presented along two dimensions: within viewpoints; and among viewpoints. Examples of tracing relationships are presented using UML. This trace model is demonstrated through its application to the design of a commercial web project using a web-design process. The design artifacts in each activity are transformed based on the artifacts tracing relationship in the trace model. The model provides mechanisms for verification of consistency, completeness and coverage within each viewpoint and the connectedness across viewpoints.

1. Introduction

In software development, it is critical to identify and understand the relationships among the artifacts of the development process such as user needs, systems requirements, design, specifications, and codes (Palmer 1996). The degree to which each element in a software development product establishes its reason for existing and the degree to which a relationship can be established between two or more products of the development process are referred to as traceability (IEEE 1990). For example, each element in a bubble chart justifies its existence by referring to the requirement that it satisfies. Similarly, the degree to which the requirements and design of a given software component match describes the traceability between requirement and design. When there is a predecessor-successor or master-subordinate relationship between two artifacts of a development process, it is especially important to trace the relationship. Traceability is typically established with the use of a trace model. A trace model is defined as *a semantic network in which nodes represent objects, stakeholders and resources among which traceability is established through links of different types and strengths* (Ramesh et al. 2001). The ability to support traceability analysis of systems artifacts is crucial in systems development and maintenance. Traceability gives essential assistance in understanding the relationships within and across software requirements, design, and implementation. Tracing provides insights to quality, consistency, completeness, impact analysis, system evolution, and process improvement (Palmer 1996).

Most existing trace models are for conventional software systems and provide little insight on how to develop trace models for web-centric systems. Compared to conventional software systems, a web-centric system typically has a tighter linkage between the business model and the technical architecture; uses technologies that change rapidly; faces high levels of client uncertainty with regard to their needs and also in terms of understanding whether a design will satisfy their needs; has high levels of change in requirements, project scope and focus, due largely to the continual evolution of the business model; and demands fine-grained evolution and maintenance with an ongoing process of content updating, editorial changes and interface tuning (Lowe et al. 2001). Web systems must cater for users of varied skill and capability (Murugesan 2000). Current web development efforts frequently bring together professionals from many varied backgrounds. These professionals have different expectations, experience and thus their contributions need to be interpreted in appropriate ways that can be understood by all project stakeholders (Powell et al. 1998; Burdman 1999).

The objective of this paper is to present a trace model specifically designed for web-centric systems based on Web Application Architecture Framework (WAAF) (Kong et al. 2005). In the next sections, literature on existing trace models is reviewed and WAAF, which forms the foundation for the trace model, is summarized. Then, the trace model is presented and illustrated through the application to the design of a commercial web project. .

2. Existing trace models

Existing software trace models tends to concentrate on two different levels. One is on source code dependency at the program level. Various trace models were developed for testing, debugging, maintenance, code optimisation and computer security purposes (Podgurski et al. 1990; O'Neal et al. 2001). Data and control dependency and program slicing are common tools for traceability analysis at this level (O'Neal et al. 2001). Example modelling approaches include feature location using dependence graphs (Chen et al. 2000), visual hierarchical approach using dependence graphs by grouping programs to extract and handle data & control dependences (Balmas 2004), and transient hypertext approach to provide support for C program maintainers (Koskinen et al. 2004).

The other is on traceability relationships at the systems level. Research has focused on analysis of traceability between all work products or documents of the development processes (O'Neal et al. 2001). Example trace models include contribution structures that consider contributions, communication and cooperation amongst the stakeholders, roles and policies (Gotel 1996), actor dependency model (Yu 1993), method-driven trace capture (Domges et al. 1998), extended traceability (Haumer et al. 1998), and a trace model for system requirement change in embedded systems (von Knethen 2002).

Some trace models cover the whole life cycle while others focus on traceability in specific areas or stages of the life cycle. For example, the V-model provides a trace model for system verification and validation at a meta-level (Sommerville 2001). Similarly, reference models for requirements traceability (Ramesh et al. 2001) and the PRIME framework (Pohl et al. 1999) concentrate on systems traceability. Commercial CASE tools have been developed to support system traceability such as DOORS (Telelogic 2005), CORE (Vitech Corporation 2005), RequisitePro and AnalystStudio (IBM

2005). These existing trace models are based on architectures for conventional software systems. The differences between web-centric systems and conventional software systems means that these trace models do not specifically address web characteristics and thus cannot be readily applied to web-centric systems.

Two models that are worth mentioning are Open Distributed Processing – Reference Model (RM-ODP) (ISO/IEC 1996a, 1996b, 1998a, 1998b) and Model Driven Architecture (MDA) (OMG 2005a). Common to both models are the focus on presenting platform independent and platform specific technologies in separate models through high level of abstraction. For example, a system that comprise Microsoft .NET, SUN's ONE and CORBA and other middleware technologies can be represented by a platform independent UML model and three platform-specific models (each for .NET, ONE and CORBA, respectively). The models describe the functionality and behaviour of distributed systems and can be used to trace concepts by correspondence. Detailed technical specification and the focus on technology limit the applicability of the two models to web projects, which typically have multi-skilled development teams, strong emphasis on user interface and driven by evolving business needs.

Most trace models are based on an underlying architectural framework. The Zachman framework (Zachman 1987) is a widely acknowledged architecture framework to classify and integrate architectures into a comprehensive matrix based on a building metaphor. A key feature of an architecture framework is its ability to trace artifacts across the entire architecture. Stakeholders in a system process are multi-disciplined and have different preferences in using modelling approaches. An architecture framework is able to establish the traceability among the different artifacts that different stakeholders have produced. The Zachman Framework matches the entities, processes, locations, people, schedule, 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. Aspects such as reusable information and an emphasis on user interface, which are typical characteristics of web systems, have not been mapped into the building metaphor.

The Web Application Architecture Framework (WAAF) (Kong et al. 2005) extends the Zachman framework to support web systems. WAAF provides traceability between the business models and technical architectures in a taxonomic way to address the characteristics of web systems – including tighter linkage between business models and technical architectures, high levels of requirements change, increased emphasis on user interfaces, and the stronger roles the design artifacts play in development. Based on WAAF, this paper presents a trace model at the system architecture level specifically designed to consider web characteristics. This trace model provides a theoretical foundation for autonomous web system code generation.

3. Web application architecture framework (WAAF)

A presentation of a whole system from the perspective of a related set of concerns is referred to as 'view'. A viewpoint establishes the conventions by which a view is created, depicted and analysed (IEEE 2000). The Web Application Architecture Framework (Kong et al. 2005) separates concerns of a web system into two dimensions. As shown in Table 1, the horizontal dimension (rows) is concerned with the different viewpoints of the stakeholders in the development process. The viewpoints include viewpoints of planners, business owners, web system users, information architects, system architects, developers and testers. The vertical dimension (columns) classifies the architectures into four abstract 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: 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 lists and describes patterns.

Table 1 WAAF Matrix - Web Application Architecture Framework

	Structure (What)	*Behaviour (How)*	*Location (Where)*	*Pattern*
Planning Architecture *(Planner's Viewpoint)*	List of things important to the business	List of processes the business performs	List of business operation locations	-
Business Architecture (Bus. Owner's Viewpoint)	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 Viewpoint)	e.g. User Interface Structure Model	e.g. User Interface Flow Model	e.g. User Site Map Model	e.g. Interface Templates, Navigation Patterns
Info. Arch. (Information Architect's Viewpoint)	e.g. Information Dictionary	e.g. Information Flow Model	e.g. Information Node Location Model	e.g. Information Scheme Patterns
System Arch. (System Architect's Viewpoint)	e.g. System Functioning Module / Server Page Structure	e.g. Workflow Model of Module / Server Page	e.g. Site Mapping of Modules Server Pages	e.g. Design Patterns, Presentation styles
Web Object Architecture (Developers' Viewpoint)	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 Viewpoint)	e.g. Test Configuration	e.g. Test Procedure	e.g. Test Deployment	e.g. Templates, Standards of Test Documents

Planning Architecture (PA, Planner's Viewpoint): concerned with issues important to the planning of the web system. Patterns are not typically relevant to this viewpoint.

Business Architecture (BA, Business Owner's Viewpoint): models the business structure, process, locations, and patterns of the web application system from the viewpoint of business owners.

User Interface Architecture (UIA, User's Viewpoint): describes the components of the system, their roles and relationships as they are perceived by users of the web system.

Information Architecture (IA, Information Architect's Viewpoint): classifies and constructs the structure, relationships, flows and location of information that are needed in the user interface viewpoint to connect the external and internal users to access the contents and the functionality of the web application. This viewpoint is independent from the implementation of the system.

System Architecture (SA, System Architect's Viewpoint): this is for design elements, including structure, behaviour, location and pattern of the web application system.

Web Object Architecture (WOA, Developer's Viewpoint): the system design is implemented into source code and other web objects. WOA describes the architecture of source code and other web objects from developer's viewpoint. Web objects are objects implemented in a web site.

Test Architecture (TA, Tester's Viewpoint): includes the structure of test documents, test procedure, location model and patterns in different types of tests.

For WAAF, each column has a unique model and each row presents a unique viewpoint. The framework does not imply the ordering of the viewpoints, nor is it necessary to provide explicit documented models for all cells. The model should support existing development paradigms.

4. WAAF-Trace, a web system trace model

The web system trace model WAAF-Trace in this paper is developed along the two dimensions in the WAAF framework. The relationships of the architectural elements are established within each viewpoint for a particular stakeholder (columns) and between different viewpoints of different stakeholders (rows) of the WAAF matrix. Below, the trace relationships along each of the two dimensions are presented using UML 2.0 notations (OMG 2005b).
A WAFF viewpoint can be modelled as a package. An abstraction is a sub-package within the package of viewpoint. Tracing relationships among packages are described by stereotyped dependencies. Figure 1 illustrated the tracing relationship between two packages. In this example, elements in client package *Cell (IA-Behaviour)* derives elements from supplier package *Cell (UIA-Behaviour).*

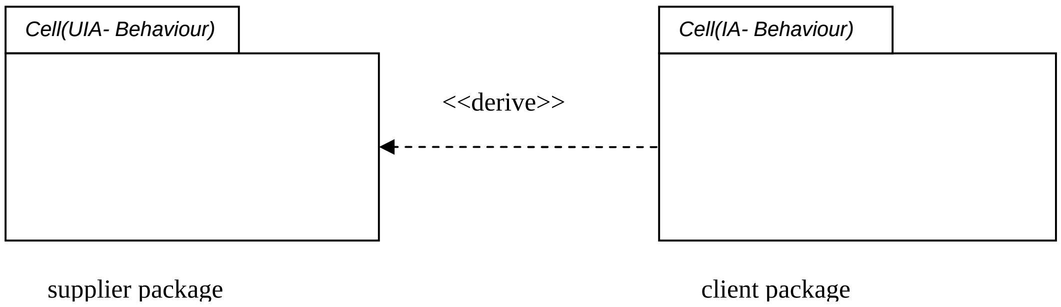


Figure 1. Example of tracing relationship model using UML stereotyped dependency.

The stereotyped dependency types used in our tracing model are described in Table 2.

Table 2. Stereotyped dependency types

Specific stereotype	**Description**
<<cluster - generate>>	Elements in supplier package are clustered and then elements of client package are generated from the clustered supplier elements.
<<define>>	Elements in suppler package are defined in client package.
<<derive>>	Client package is derived from supplier package.
<<generate>>	Elements in supplier package are generated from elements in client package.
<<model>>	Elements listed in supplier package are abstracted into models in client package.
<<present>>	Elements in supplier package are presented in user interfaces in package 'User Interface Architecture'.
<<realize>>	Models in supplier package are realized in client package.
<<refine>>	Models in supplier package are further refined to a more detailed level in the client package.
<<refine – model>>	Elements in supplier package are refined to a more detailed level and abstracted into models in the client package.
<<refine – specify>>	Elements in supplier package are refined to a more detailed level and the interrelationships are specified in the client package.
<<satisfy>>	Elements in the client package are used to satisfy the needs of elements in the supplier package.
<<specify>>	Elements or relationships that are listed in the supplier package are specified in the client package.
<<transfer>>	Behaviour models in the supplier package are transferred into the client package.
<<validate>>	Elements in the supplier package are validated by elements in the client package.
<<verify>>	Elements in the supplier package are verified by elements in the client packages.

The entire trace model for a web system is an integration of the tracing relationships along two orthogonal dimensions, namely *abstract* and *viewpoint*. In the sections below, tracing using the WAAF-Trace model is described.

4.1 Trace model within viewpoints – column dimension

Each row in the WAAF Matrix presents a unique viewpoint. Each viewpoint describes an entire model for a stakeholder. The traceability within a viewpoint can be modelled among the abstractions of structure, behaviour, location and pattern. For simplicity, the illustrations below only model the traceability between structure and behaviour.

The following trace relationships exist within a viewpoint dimension:

The *behaviour model* describes the functioning workflow process of the system from a stakeholder's viewpoint. The workflow process models the interactions of the *entities* comprising the system. The *entities* in the *behaviour model* are described and their relationships are modelled in the *structure model*.

An example of the trace relationship is demonstrated in Figure 2. The example is based on the 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". The stakeholders of "XYZ-Match" adopted agile modelling approaches in their development lifecycle. Models in different stakeholders' viewpoints are presented in different modelling "languages". For example, structure models are described by entity-relation diagrams. Behaviour models are flow charts, user interface flow diagrams, information flow diagrams and server page flow diagrams.
(Note: In the following XYZ-match models, entities in each viewpoint are drawn in a box. Entity types are represented by <<>>. Entity names are listed under the entity types.)

Figure 2 shows the structure model and the behaviour model of a User Interface (UI) viewpoint. In the behaviour model, a workflow of an entrepreneur matching business proposal to Venture Capital (VC) firm investment requirements is modelled as a UI diagram in Figure 2(b). The UI diagram shows the navigation flow of the user interfaces "Entrepreneur_home", "edit_entrepreneur_details", "submit_biz_proposal", "Match_VCFirm_Entrepreneur_Form", "List_of_VCFirms", and "VCFirm_details". These user interfaces in *Cell(UI-Behaviour)* and their interrelationships are modelled as an entity-relation diagram in *Cell(UI-Structure)* in Figure 2(a).

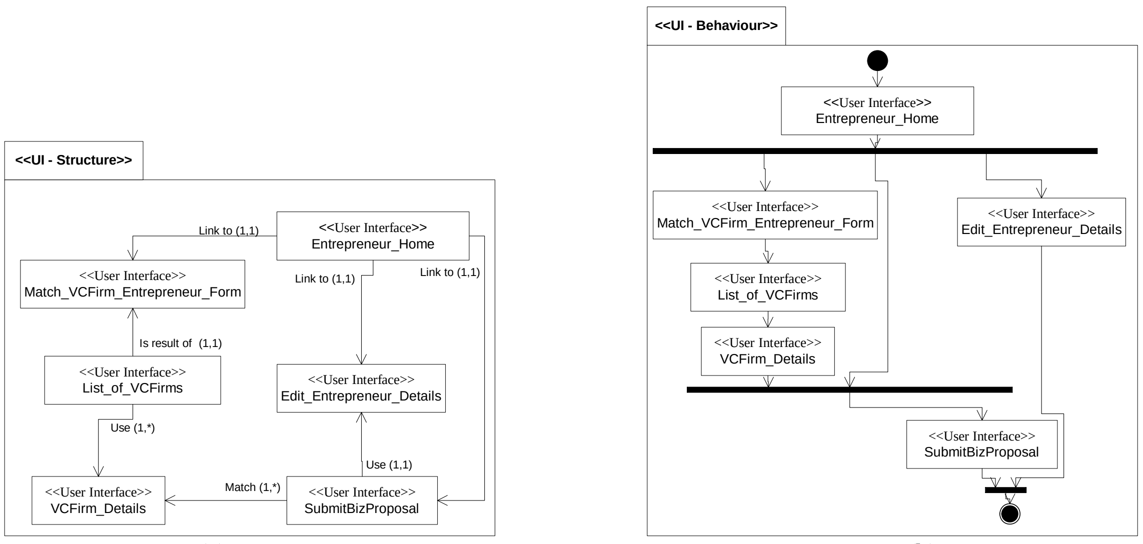

<<UI - Structure>>
Link to (1,1)
<<User Interface>> Entrepreneur_Home
<<User Interface>> Match_VCFirm_Entrepreneur_Form
Link to (1,1)
Link to (1,1)
Is result of (1,1)
<<User Interface>> List_of_VCFirms
<<User Interface>> Edit_Entrepreneur_Details
Use (1,*)
Use (1,1)
<<User Interface>> VCFirm_Details
Match (1,*)
<<User Interface>> SubmitBizProposal
<<UI - Behaviour>>
<<User Interface>> Entrepreneur_Home
<<User Interface>> Match_VCFirm_Entrepreneur_Form
<<User Interface>> Edit_Entrepreneur_Details
<<User Interface>> List_of_VCFirms
<<User Interface>> VCFirm_Details
<<User Interface>> SubmitBizProposal
(a) (b)

Figure 2. User Interface viewpoints
(a) example of *Cell(UI-Structure)* (b) example of *Cell(UI–Behaviour)*

4.2 Trace model between viewpoints – row dimension

In the WAAF Matrix, PA and BA are business architectures and UIA, IA, SA, WOA and TA are web system architectures. 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 traced into software architectures.

Design artifacts play a significant role in web development (Lowe et al. 2003) and could lead to necessary changes in the business architecture. Artifacts in technical architectures and business architecture are mapped into each other. WAAF allows forward and backward tracing among viewpoints. Using the existing order in the WAAF Matrix, the tracing relationships between each two viewpoints can be modelled as follows:

4.2.1 Planning Architecture (PA) & Business Architecture (BA)

The tracing relationship of PA and BA is illustrated in Figure 3.

- *Cell (BA-Structure)* in the Business Architecture refines and specifies the relationships among the business entities that are listed in *Cell (PA-Structure*) of the Planning Architecture.
- *Cell (BA-Behaviour)* in the BA models the business processes listed in the PA in Cell *(PA-Behaviour).* One business process in PA could be refined into a number of business processes in BA.

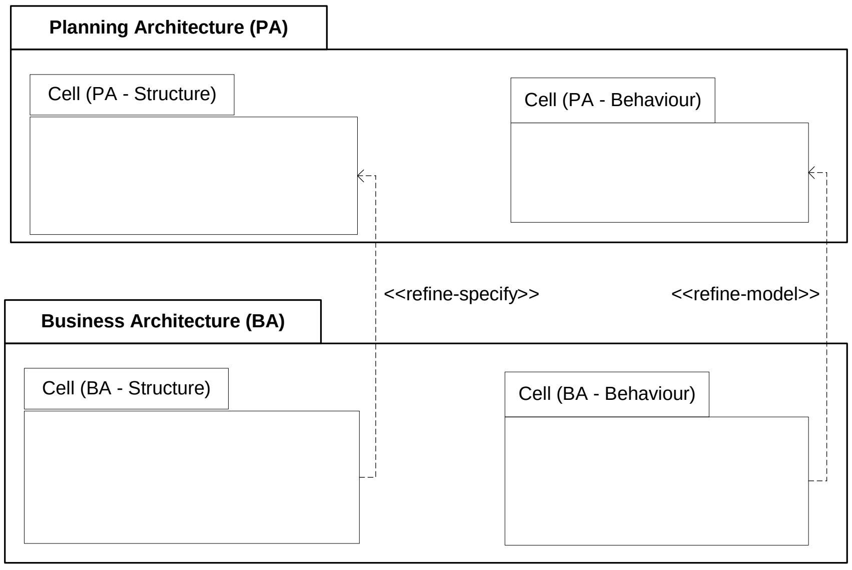


Figure 3. Tracing relationship of PA and BA

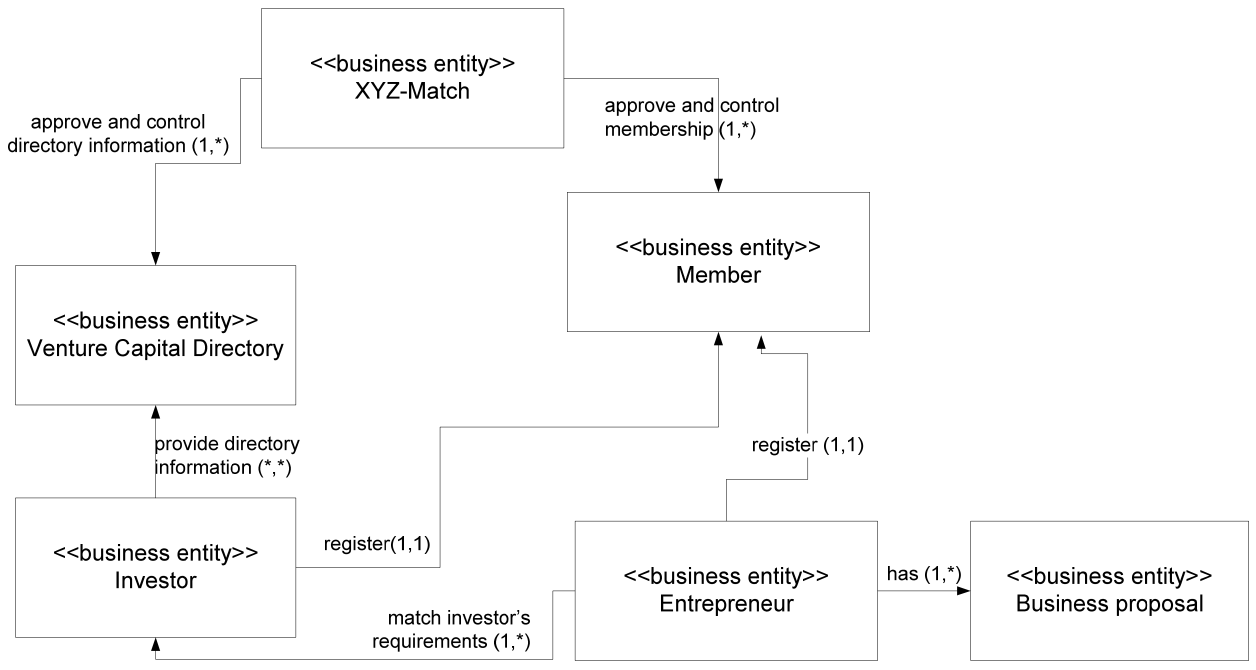
This tracing relationship is demonstrated using an example of XYZ-Match.

In *Cell (PA-Structure)*, the following 3 systems/entities that are important to the business model are listed:

- XYZ-Match web system

- Venture capital directory
- XYZ-Match members

In *Cell (BA-Structure)*, these systems/entities are refined to 6 major business entities: XYZ-Match, venture capital directory, member, investor, entrepreneur and business proposal. The relationships of the major business entities are specified in an entity-relation diagram in Figure 4.



<<business entity>> XYZ-Match
approve and control directory information (1,*)
approve and control membership (1,*)
<<business entity>> Member
<<business entity>> Venture Capital Directory
provide directory information (*,*)
register (1,1)
<<business entity>> Investor
register(1,1)
match investor's requirements (1,*)
<<business entity>> Entrepreneur
has (1,*)
<<business entity>> Business proposal

Figure 4. Business structure of XYZ-Match

In *Cell (PA-Behaviour)*, the 2 major processes of XYZ-Match are listed:

- XYZ-Match web system provides venture capital directory.
- Members use XYZ-Match system for venture capital matching.

In *Cell (BA-Behaviour)*, these 2 major processes can be further refined into the following 4 major business processes:

- Users register in “XYZ-Match” to become a “member”.
- “Investors” provide venture capital information in “Venture capital directory”.
- “Entrepreneurs” search for venture capital.
- “Entrepreneurs” and “investors” match in “XYZ-Match” web system.

Each process is modeled in *Cell (BA-Behaviour)* using dynamic diagrams such as flow chart, activity diagram, etc. Business process models in *Cell (BA-Behaviour)* can be refined further if necessary.

4.2.2 Business Architecture (BA) & User Interface Architecture (UIA)

Web system users interact via a web interfaces with business entities that are relevant to the web system. Web components in the BA are mapped into web interfaces as users' views. Views of users in the UIA are technology independent. As shown in Figure 5, the tracing relationships are as follows:

- *Cell (UIA-Behaviour)* in the UIA transfers the business processes models in *Cell (BA-Behaviour)* to user interface flow models.
- *Cell (UIA-Structure)* in the UIA presents the user interfaces and their relationships for web-system-relevant business entities described in *Cell (BA-Structure*) in the Business Architecture.

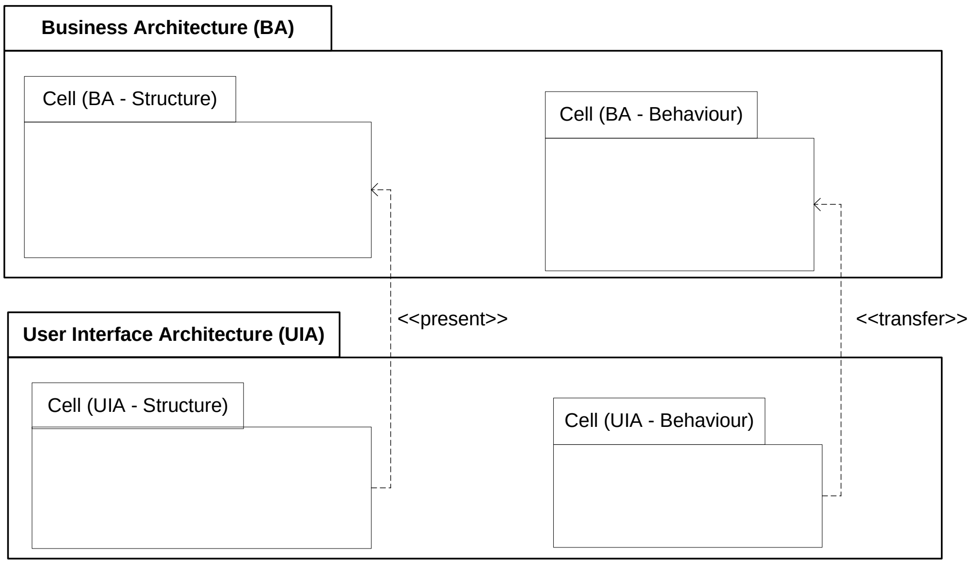
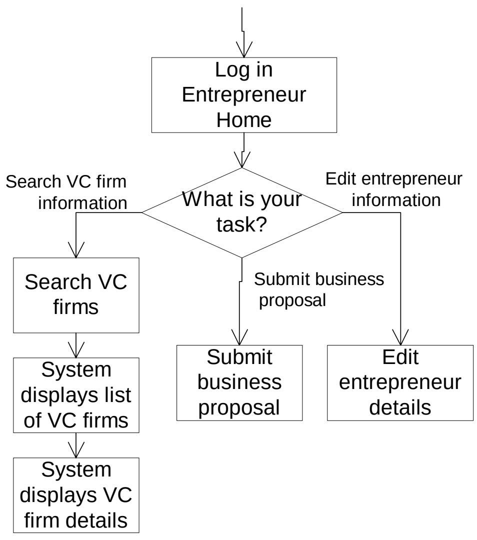


Figure 5. Trace model between BA and UIA

Figure 6 shows a business flow model in *Cell (BA-Behaviour)*, drawn in a flow chart. In *Cell (UIA-Behaviour)*, this business flow is transferred into a User Interface diagram as illustrated in Figure 2(b). In the User Interface diagram, each user interface is traced to each activity step in the business flow diagram. For example, "Match_VCFirm_Entrepreneur_Form" in the User Interface diagram in Figure 2(b) is the user interface for the activity step "Search VC firms" in the business flow model in Figure 6.



Log in Entrepreneur Home
What is your task?
Search VC firm information
Edit entrepreneur information
Submit business proposal
Search VC firms
System displays list of VC firms
System displays VC firm details
Submit business proposal
Edit entrepreneur details

Figure 6. Example of a business flow model in *Cell(BA-Behaviour*)

Figure 2(a), in turn, presents the user interfaces that described in *Cell(BA-Structure)*.

4.2.3 User Interface Architecture (UIA) & Information Architecture (IA)

Information relating to the handling and support of User Interface is modelled under User Interface Architecture.

- *Cell (IA-Structure)* in the IA structures, organises and labels the information that needed to be presented to the user interfaces in *Cell (UIA-Structure)* in the User Interface Architecture. This information depends-on and satisfies the need of user interfaces in UIA.
- *Cell (IA-Behaviour)* in the IA derives the user interface flows in *Cell (UIA-Behaviour)* of UIA into information flows to model the information creation, exchange, process and consumption that support the user interface flows.

Figure 7 shows the relationships between the IA and the UIA.

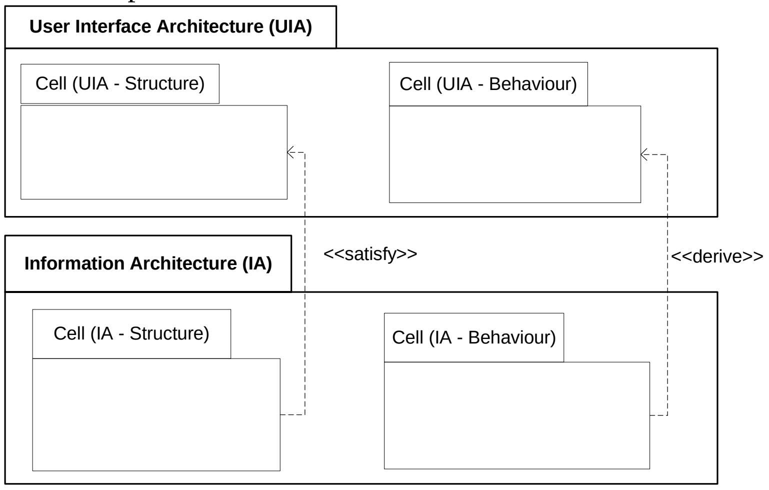
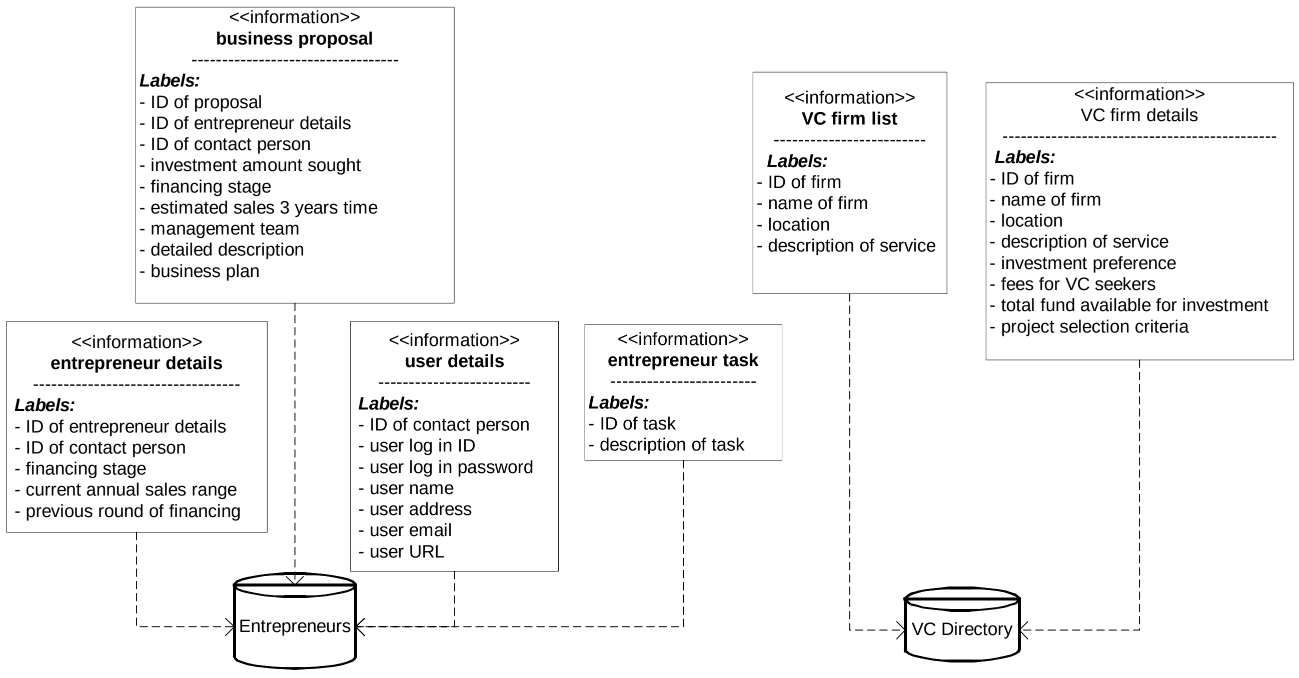


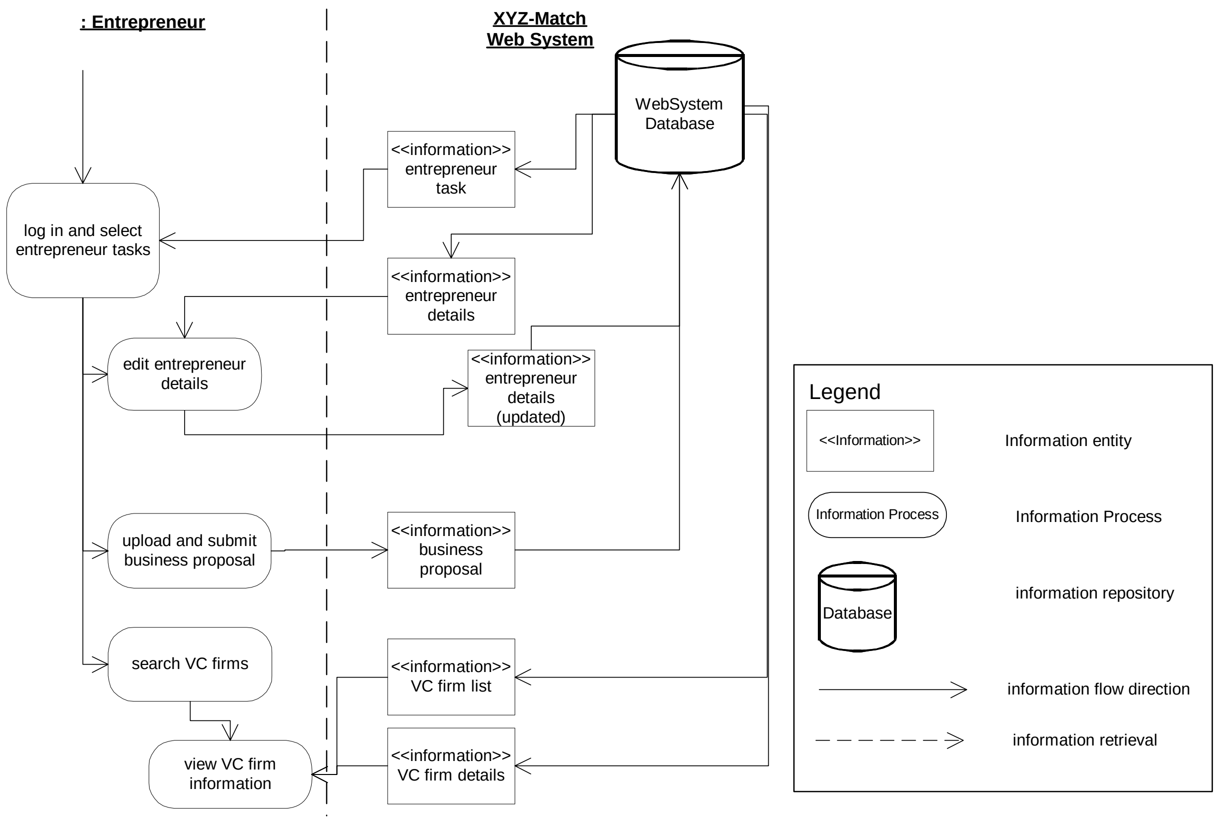
Figure 7. Trace model between IA and UIA

Figure 8(a) is an information structure for organizing and labelling the information for the user interfaces in Figure 2(a). In the information structure, information entities “business proposal”, “entrepreneur details”, “user details”, “entrepreneur task”, “VC firm list”, and “VC firm details” are described. These information entities are needed for user interfaces shown in Figure 2(a). Within each information entity, information labels are listed. Each information entity is retrieved from an information resource as shown in Figure 8(a). The dash line, which starts from an information entity and ends to an information resource, shows the information retrieval relationship. For example, information entity “VC firm details” is retrieved from information source “VC Directory”. Information entity “business proposal” is retrieved from “Entrepreneurs”.

Figure 8(b) is an information flow diagram derived from Figure 2(b). This diagram shows the flow of information process that supports the user interface flow in Figure 2(b).



(a)


: Entrepreneur
XYZ-Match Web System
WebSystem Database
<<information>> entrepreneur task
log in and select entrepreneur tasks
<<information>> entrepreneur details
edit entrepreneur details
<<information>> entrepreneur details (updated)
upload and submit business proposal
<<information>> business proposal
search VC firms
<<information>> VC firm list
view VC firm information
<<information>> VC firm details
Legend
<<Information>>
Information entity
Information Process
Information Process
Database
information repository
information flow direction
information retrieval

(b)

Figure 8. (a) An example of information structure. (b) An example of information flow diagram in *Cell (IA-behaviour*) derived from user interface diagram

4.2.4 System Architecture (SA), & Information Architecture (IA) & UIA

The inputs of the System Architecture are from UIA and IA.

- *Cell (SA-Structure):* User interfaces in *Cell (UIA- Structure*) are clustered into modules and refined into sub modules in *Cell (SA-Structure).* Within submodules, server pages (source code) are included to realise the user interfaces in *Cell (UIA- Structure*) and generate the information (or content) that are described in *Cell (IA- Structure*).
- *Cell (SA-Behaviour)* in the SA realises the process in information flows in *Cell (UIA – behaviour)* to server page flows. Server page flows generate the user interfaces in user interface flow diagrams in *Cell (UIA-Behaviour)* and the information (content) in the information flows in *Cell (IA-Behaviour).*

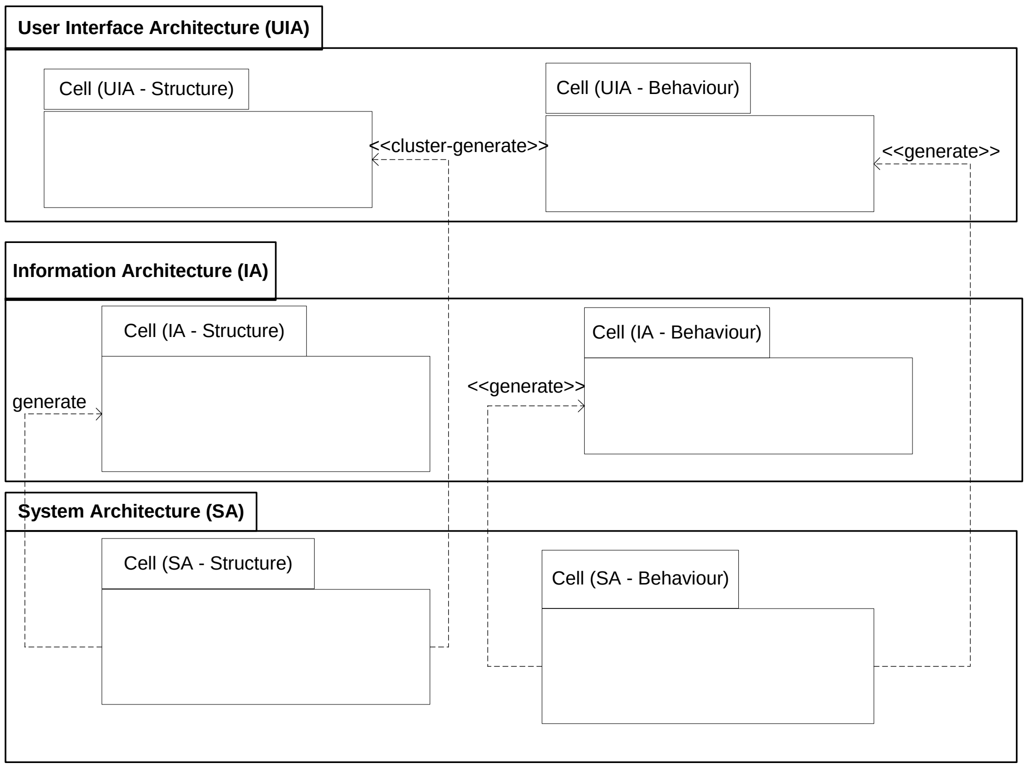


Figure 9. Trace model for SA to UIA and IA

As shown in Figure 9, user interfaces in *Cell(UIA-Structure*) that are related to investors and entrepreneurs are clustered into module "VC Directory" and "Investment Opportunities" in *Cell(SA-Structure*) respectively. Within "Investment Opportunities" module, a number of submodules are defined. Submodule "entrepreneurs submit business proposals" includes a set of server pages to realise the user interfaces for business proposal submission, and to generate information entities "business proposal", "entrepreneur details", and "user details" which are described in information structure in Figure 8(a). Three server pages "GetInfo_EntrepreneurDetails.cfm", "Edit_EntrepreneurDetaislForm.cfm", and "UpdateInfo_EntrepreneurDetails.cfm" are used to generate the information entity "entrepreneur details" in Figure 8(a) and the user interface "edit_entrepreneur_details" in Figure 2(b).

Figure 10 shows an example of the trace relationship among the server page flow diagram in *Cell (SA-Behaviour),* the user interface diagram in *Cell (UI-behaviour),* and the information flow diagram in *Cell (IA-Behaviour).* The sequence of server pages "GetInfo_EntrepreneurDetails.cfm", "Edit_EntrepreneurDetaislForm.cfm", and "UpdateInfo_EntrepreneurDetails.cfm" are defined to generate the user interface flow and the information process flow for editing entrepreneur details.

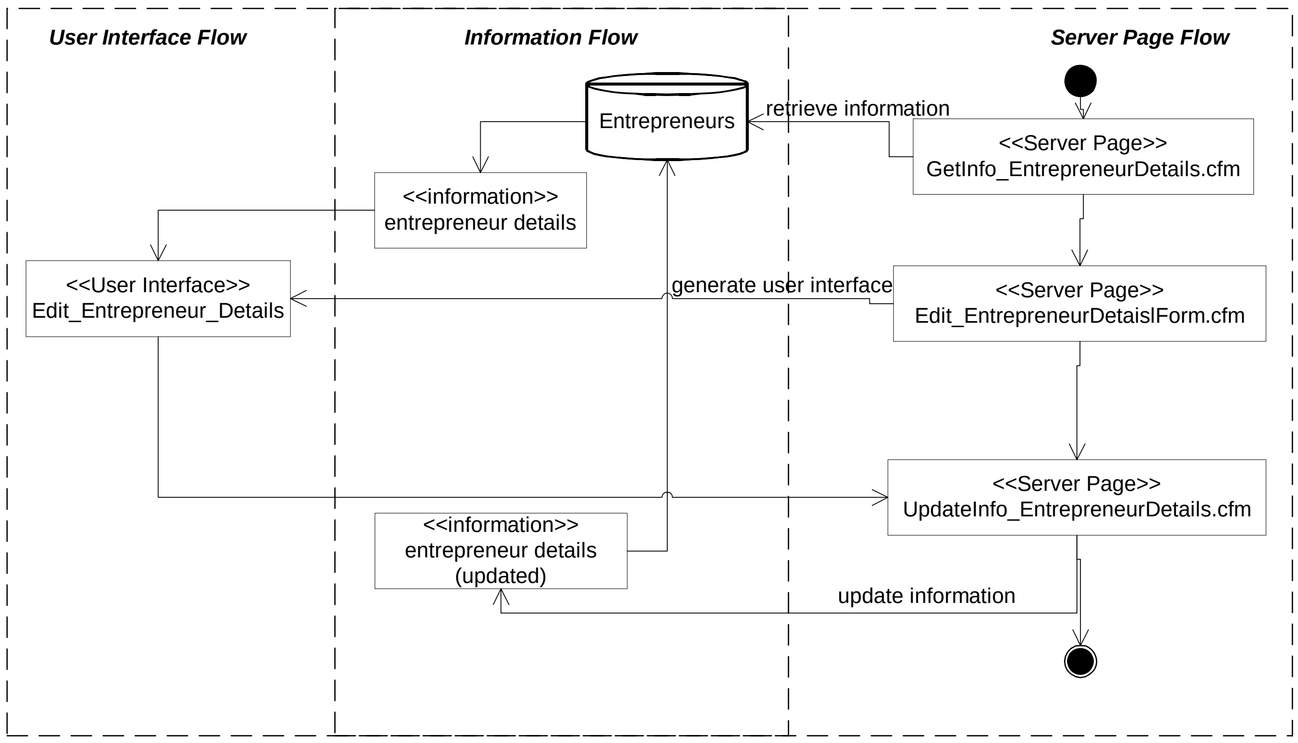

User Interface Flow
Information Flow
Server Page Flow
Entrepreneurs
retrieve information
<<Server Page>> GetInfo_EntrepreneurDetails.cfm
<<information>> entrepreneur details
<<User Interface>> Edit_Entrepreneur_Details
generate user interface
<<Server Page>> Edit_EntrepreneurDetaislForm.cfm
<<Server Page>> UpdateInfo_EntrepreneurDetails.cfm
<<information>> entrepreneur details (updated)
update information

Figure 10. Example of the trace relationship among *Cell (SA-Behaviour), Cell (UI-behaviour)*, and *Cell (IA-behaviour)*.

4.2.5 Web Object Architecture (WOA) & System Architecture (SA)

The System Architecture is implemented into a Web Object Architecture.

- *Cell (WOA-Structure)* defines the input/output data or information for server pages (or source code) that are listed in *Cell (SA-Structure)*. The relationships of web objects that will be used in source code are specified. The database scheme to store contents used in server pages (or source code) is defined in *Cell (WOA-Structure).*
- *Cell (WOA-Behaviour)* refines the server page flow in *Cell (SA-Behaviour)* into detailed algorithms in each server page.

The trace model between WOA and SA are illustrated in Figure 11.

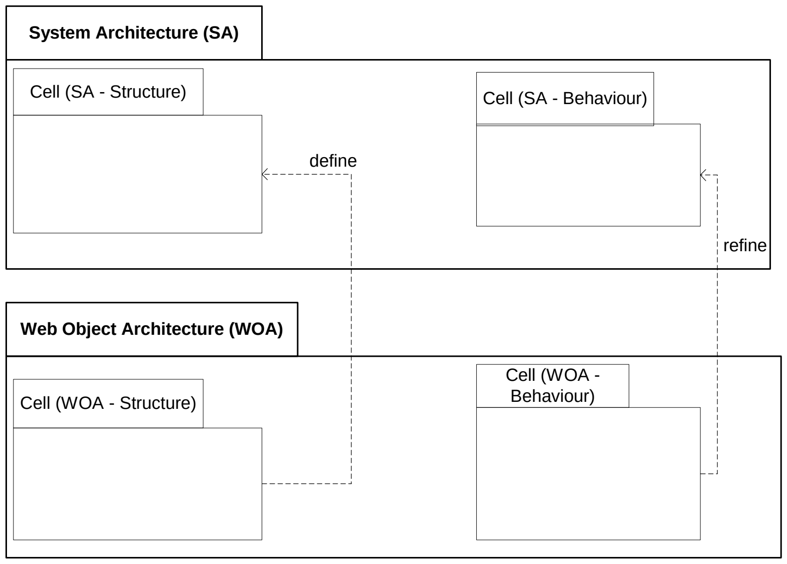


Figure 11. Trace model between System Architecture and Web Object Architecture

Figure 12(a) provides an example of *Cell* (*WOA-Structure*). The input and output of server page 'GetInfo_EntrepreneurDetails.cfm' are defined. Server page 'GetInfo_EntrepreneurDetails.cfm' is listed in *Cell* (*SA-Structure*). The server page flow of *Cell (SA-Behaviour)* in Figure 10 is refined into the algorithm as illustrated in Figure 12(b). Here, we only provide one illustrative algorithm for the server page 'GetInfo_EntrepreneurDetails.cfm'.

Cell(*WOA-Structure*)	*Cell*(*WOA-Behaviour*)
<<Server Page>> GetInfo_EntrepreneurDetails.cfm --------------------------------------- Input: 'Current_entrepreneur_ID' Output: 'Information_of_current_entrepreneur': - financing stage - current annual sales range - previous round of financing - user name - user address - user email - user URL	<<Server Page>> Algorithm GetInfo_EntrepreneurDetails.cfm --------------------------------------- Select 'Information_of_current_entrepreneur'; From information resource 'Entrepreneur'; Where Entrepreneur_ID is Current_entrepreneur_ID; Display 'Information_of_current_entrepreneur' in user interface 'Edit_Entrepreneur_Detailsform.htm'.
(a)	(b)

Figure 12. Example of (a) *Cell(WOA-Structure*) and (b) *Cell(WOA-Behaviour*)

4.2.6 Test Architecture (TA) & all other architectures

Testing includes verification and validation of the artefacts produced in the rows above the TA viewpoint in the WAAF Matrix, and seeking the weak points of the artifacts produced from Web Object Architecture. The traceability from the Test Architecture should therefore relate to all other viewpoints to verify the satisfaction of the system against aspects such as the business architecture, information architecture, and web object architecture.

- *Cell (TA-Structure)* defines the test documents such as the test plan, items under test, test design, test case, test data, test procedure, test incident report and test summary report to verify the planning, business architecture, user interface architecture, information architecture, system design architecture and web object architecture respectively.
- *Cell (TA-Behaviour)* defines the procedures or flows to verify and validate the above viewpoints.

Figure 13 shows how the TA traces to other architectures.

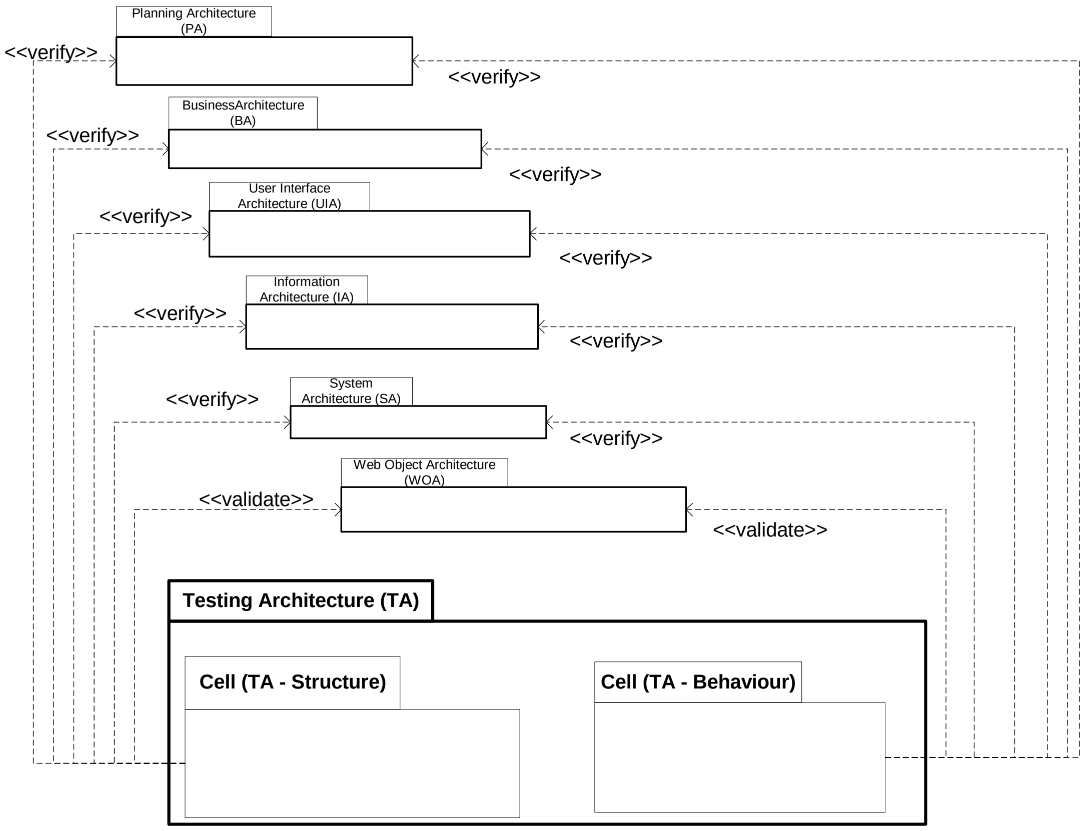


Figure 13. Trace model from TA to PA, BA, UIA, IA, SA and WOA

4.3 Summary of the WAAF-Trace model

- In the column dimension (abstractions), the WAAF-Trace model provides a tool for verification of consistency within each viewpoint. For example, the *structure model* lists entities that are needed for a stakeholder's viewpoint. The *behaviour model* describes the interaction of these entities. The trace relationship between the *structure model* and the *behaviour model* determines the completeness and coverage of entities from the *structure model* to the *behaviour model.*

- The row dimension (viewpoint) provides a tracing mechanism to verify the needs (or requirements) of each viewpoint inherited from other viewpoints.
- The multi-dimensional tracing within and among viewpoints provides a tool for impact analysis of web systems that have characteristics such as fine-grained evolution, maintenance and with a high level of requirements, project scope and focus change.

To further demonstrate how the WAAF-Trace model can be applied, we apply it to the design process of a commercial web site in the next section.

5. WAAF-Design, a design method based on the WAAF-Trace model

In this section, the application of WAAF-Trace model is demonstrated through step-by-step design of a commercial web project. The design process also used a design-process called WAAF-Design. Below, the designed process is briefly introduced first. Then, the application of the trace model to the designed process is described.

5.1 Background

Over the last decade numerous approaches to web system design have emerged. Early approaches evolved from Entity-Relationship modelling in the Hypertext community including structured analysis and design in the information domain, and object-oriented analysis design. RMM (Isakowitz 1995) provides a structured design methodology for hypermedia applications. Its focus is on modelling the underlying content, the user viewpoints on this content, and the navigational structures that interlink the content. OOHDM (Schwabe 1998) is a similar approach based on object-oriented software modelling, though somewhat richer in terms of the information representations. Other similar design methods include EORM (Lange 1994) and work by Lee (1997). WSDM (Troyer et al. 1997) attempts to take these approaches one step further, by beginning more explicitly from user requirements. In general, these techniques were either developed explicitly for modelling information in the context of the Web, or simply adapted from conventional software domain to the Web domain. More recently, work on WebML has begun to amalgamate these concepts into a rich modelling language for describing Web applications. The focus of WebML is very much on content modelling rather than describing the functionality that is a key element of most current commercial Web systems. One of the few approaches that attempt to integrate content representation with functionality is (Takahashi et al. 1997).

Research results have shown that in commercial web development, clients' understanding of their needs evolve as a system evolves and thus design artefacts play a crucial role in the development of the clients' understanding (Lowe 2003). A typical web development process includes a prototyping phase. User interfaces (UI) are prototyped before the system design commences (Peters et al. 2002; Lowe et al. 2003; Fusebox 2005). Agile development methodologies, which are very popular in the Web development community, emphasize the value of communication between the clients and the developers. Web system prototyping is commonly adopted as a vehicle for such communications. Web systems place a strong emphasis on user interface. Significant input is made into the "look and feel" of a web site. Usability is of paramount importance and there is considerable artistic input into page appearance. Fusebox (Peters et al. 2002; Lowe et al. 2003; Fusebox 2005) suggests that a user is not concerned with any part of the web system apart from the front end. What happens behind the front-end is of no direct concern to the users. The basic requirement for the back end is to support all the functionalities initially mocked up in the front end "prototype".

The existing web design approaches such as RMM, OOHDM and WebML run counter to the emphasis on user interface the and prototyping from the common web development processes. The user interfaces of their presentation design are typically derived from the design.

The trace model WAAF-Trace provides a streamlined design approach to address the commercial web development practice. Here we apply the WAAF-Trace to a new web design method, called "WAAF-Design". This design method addresses the design process by placing system design after the user interface prototyping. The system design artifacts are derived from the user interface prototype. WAAF-Design imposes the traceability of WAAF-Trace to a modified version of Flip (Peters et al. 2002; Fusebox 2005).

The FLiP process consists of a number of steps: Personas and goals; Wireframe; Prototype or Front-end development; Architecting; Fusecoding; Unit testing; Application Integration; Deployment. "Personas and goals" identify web application users and their goals. "Wireframe" models the proposed actions that will be performed by the application. A "prototype" of FLiP is a clickable model of the finished application with no backend behind it. In prototyping step, what the client expects from the application is revealed. Once the prototype is finished, the application architects construct the application design in step "Architecting". The architects identify fuseactions and organize them into circuits. Each fuseaction behaviour is broken down into a set of fuses (code), the architects write a Fusedoc and a test harness for each fuse. In step "Fusecoding", the coders write the fuses according to its Fusedoc. As each fuse is coded, it is unit tested against its test harness in step "Unit testing". The architects integrate completed fuses into the final application in "Application Integration" step. The prototype is gradually transformed into a working application based on daily builds. The final product is deployed from the testing server or the staging server to the production server in step "Deployment" (Fusebox 2005). FLiP is platform dependent process and therefore is not a methodology for all types of web development. In addition, developers also need tools such as "DevNotes" for prototyping and "Mind Mapping" or "Fuseminder" for site construction, to support the development. Nevertheless, the most visible deficiency in FLiP is its lack of modelling methodology comparing with other common design methodologies. As software development continue to grow in complexity, and developers are used to work at high levels of abstraction to cope with this complexity, modeling software is – and will continue to be – a key aid to developers to work at high levels of abstraction (Cernosek 2004). Without an explicit modelling method, it will be difficult to trace artifacts throughout the development process.

The WAAF-Design method aims to enhance the modelling and architecting ability and design traceability to the commercial web design process. It focuses on the design phase of the web development life cycle since this a major point of departure from conventional software development. The methodology is platform-independent and does not require any particular tools. The process includes the following steps: Analysis; Prototype; Information Modelling; Architecting; Implementation and Deployment. The "Analysis" step is equivalent to "Personas and goals, and Wireframe" in FLiP. "Implementation and Deployment" corresponds to "Fusecoding, Unit testing, Application Integration and Deployment" in FLiP. The modification of the process from FLiP is in the following 3 steps: Prototype; Information Modelling and Architecting. In Web system design, "content is the King of the web" (Lowe et al. 2003). Accordingly, information modelling is a treated as a critical step in the design process. It could be iteratively applied until a satisfactory outcome has been achieved. The design method is discussed in detail in the following section.

Table 3. Development process

Process	FLiP process	Modified Process
Step	Personas & goals; Wireframe;	Analysis
	Prototype; Architecting;	Prototyping; Information modelling; System architecting;
	Fusecoding; Unit testing; Application integration; Deployment	Implementation and deployment

5.2 WAAF-Design Methodology

The key steps of the WAAF-Design development process are presented in Table 3 in contrast with the FLiP process. The focus of this design methodology is the design stage including 3 steps: prototyping; information modelling; and system architecting.

The development is an iterative process. The transformation of the development artifacts is based on the trace relationship of the WAAF-Trace model. The artifacts of the design are initially transformed from the analysis step. At "analysis" step, the analysts and the client explore the business process for the web application being built. Potential application users and their goals are identified. Tasks to achieve the goals are discovered. Tasks are clustered to scenarios through their semantic cohesions. The artifacts of this step include list and relationship of the user groups and their goals; list of the tasks and the description of their flows; list of the scenarios. The artifacts of analysis step are then transformed into design activities. The design artifacts will be transformed into implementation step.

Figure 14 illustrates the activities and associated artifacts in the 3 steps of the design process. Each step is discussed below.

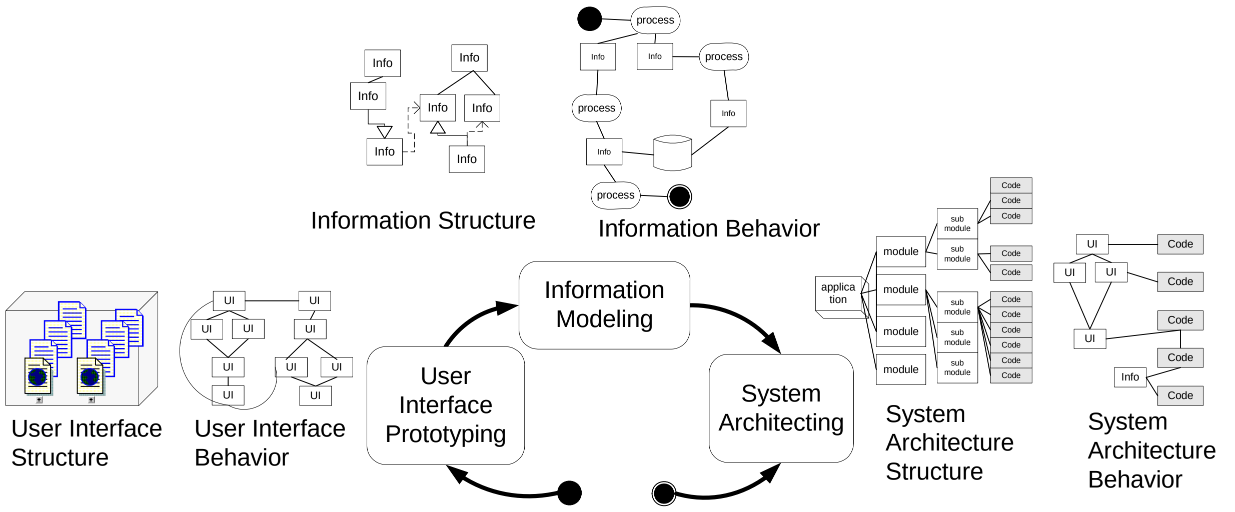


Figure 14. Activities and artifacts of design steps

5.2.1 Prototyping Step

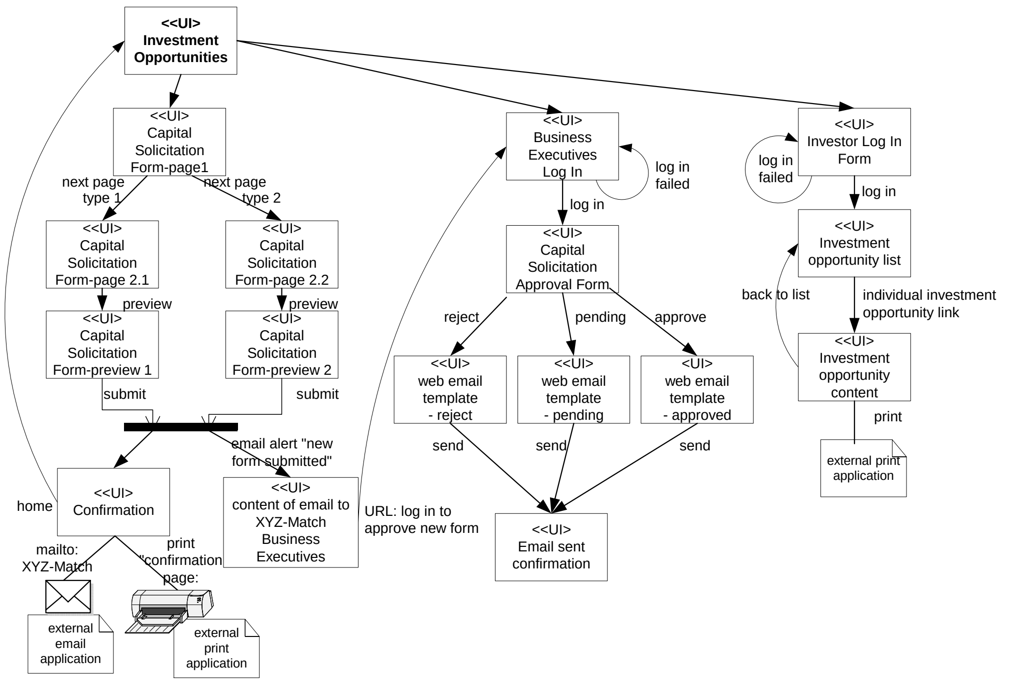
A prototype is a full-scale front-end model of a web application. It is web users' viewpoint. Prototyping step formulizes the front-end presentation of what need to be built. We consider the structure and behaviour dimension of the front-end model at prototyping step. The structure cell and behaviour cell of the User Interface Architecture in WAAF matrix are applied as two basic components for prototyping.

The behaviour of the prototype models the user navigation path to access and operate the application. The navigation paths are derived from the business process scenarios described in the analysis step. The behaviour can be modelled using user interface flow diagram (Ambler 2004) which considers the flow and state transition of user interfaces.

All the user interfaces on the navigation paths are collected in the prototype structure. The structure of the prototype consists of the collection of the user interfaces in the entire application and the layout and the content of each user interface.

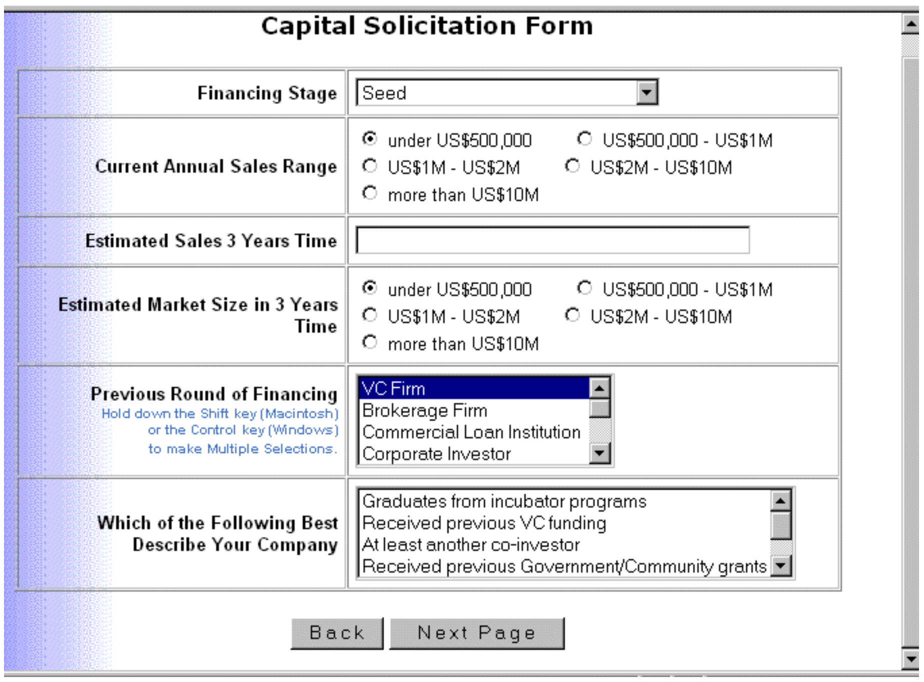
To demonstrate the design method, we use examples of the web application of "XYZ-Match". The prototyping process is illustrated using a "capital solicitation" scenario of XYZ-match. In this scenario, entrepreneurs submit business proposals and the XYZ-match approves the proposals. The whole prototype behaviour of the application will include all the business process scenarios of XYZ-match described in the analysis step.

The navigation paths of this scenario and the individual user interfaces to present these paths are identified and modelled in a user interface flow diagram in Figure 15. In UI flow diagram, each UI is represented using a box which only includes the name of the UI and exit points. The contents of each UI are presented in the prototype structure, not considered in the prototype behaviour. Figure 16 shows the content on one page on the navigation path. The contents are what the user will see in the final product.



<<UI>
Investment Opportunities
<<UI>
Capital Solicitation Form-page1
next page type 1
next page type 2
<<UI>
Capital Solicitation Form-page 2.1
<<UI>
Capital Solicitation Form-page 2.2
preview
preview
<<UI>
Capital Solicitation Form-preview 1
<<UI>
Capital Solicitation Form-preview 2
submit
submit
email alert "new form submitted"
home
<<UI>
Confirmation
<<UI>
content of email to XYZ-Match Business Executives
mailto: XYZ-Match
print "confirmation page:
external email application
external print application
URL: log in to approve new form
<<UI>
Business Executives Log In
log in failed
log in
<<UI>
Capital Solicitation Approval Form
reject
pending
approve
<<UI>
web email template - reject
<<UI>
web email template - pending
<UI>
web email template - approved
send
send
send
<<UI>
Email sent confirmation
<<UI>
Investor Log In Form
log in failed
log in
<<UI>
Investment opportunity list
back to list
individual investment opportunity link
<<UI>
Investment opportunity content
print
external print application

Figure 15. User interface flow diagram


Capital Solicitation Form

Financing Stage	Seed
Current Annual Sales Range	[✓ under US$500,000] US$500,000 - US$1M US$1M - US$2M US$2M - US$10M more than US$10M
Estimated Sales 3 Years Time	
Estimated Market Size in 3 Years Time	[✓ under US$500,000] US$500,000 - US$1M US$1M - US$2M US$2M - US$10M more than US$10M
Previous Round of Financing Hold down the Shift key (Macintosh) or the Control key (Windows) to make Multiple Selections.	[✓ VC Firm] Brokerage Firm Commercial Loan Institution Corporate Investor
Which of the Following Best Describe Your Company	Graduates from incubator programs Received previous VC funding At least another co-investor Received previous Government/Community grants

Figure 16. User interface structure

5.2.2 Information Modelling Step

Information is "the interpretation of data within a context set by a priori knowledge and the current environment" (Lowe et al. 1999). In information modelling step, information architects model the prototype into information architecture (IA). We consider two cells of WAAF Matrix *information structure* and *information behaviour* as two basic components for information modelling.

Information architects classify and construct the structure and the relationship of the information in *information structure*. The information being modelled depends-on and satisfies the need in the UI contents in the prototype.

For example, the information needed in the capital solicitation scenario of XYZ-match can be categorized into user details, entrepreneur details, VC seeking criteria, publishing permission selection, etc. Under the category "user details", the information can be labelled as "user log in ID, user log in password, user name, user address, user email, user URL" etc. (See Figure 17).

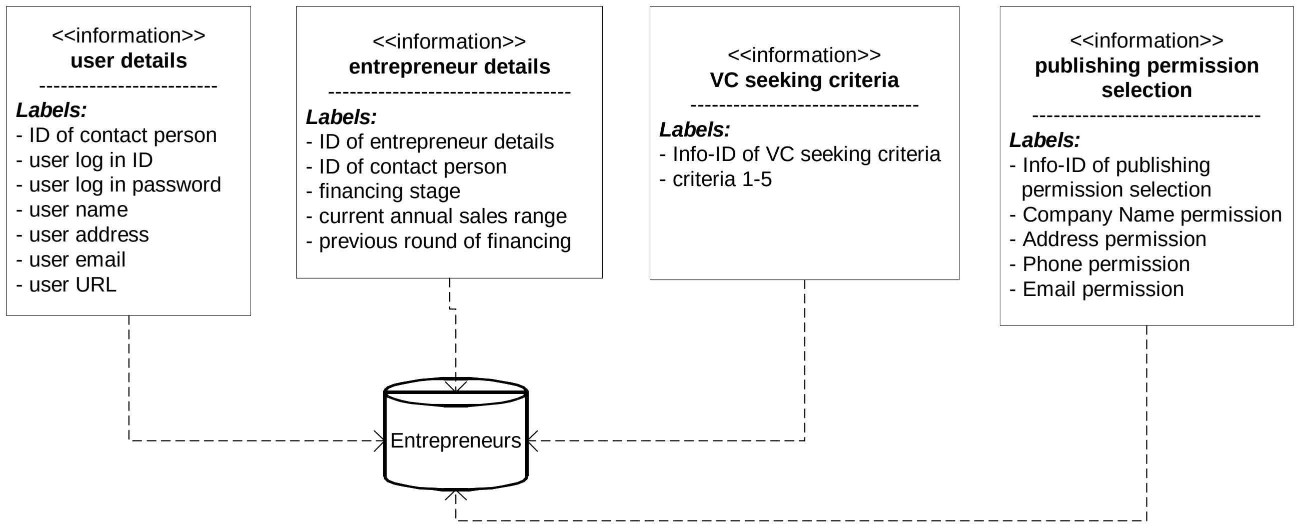
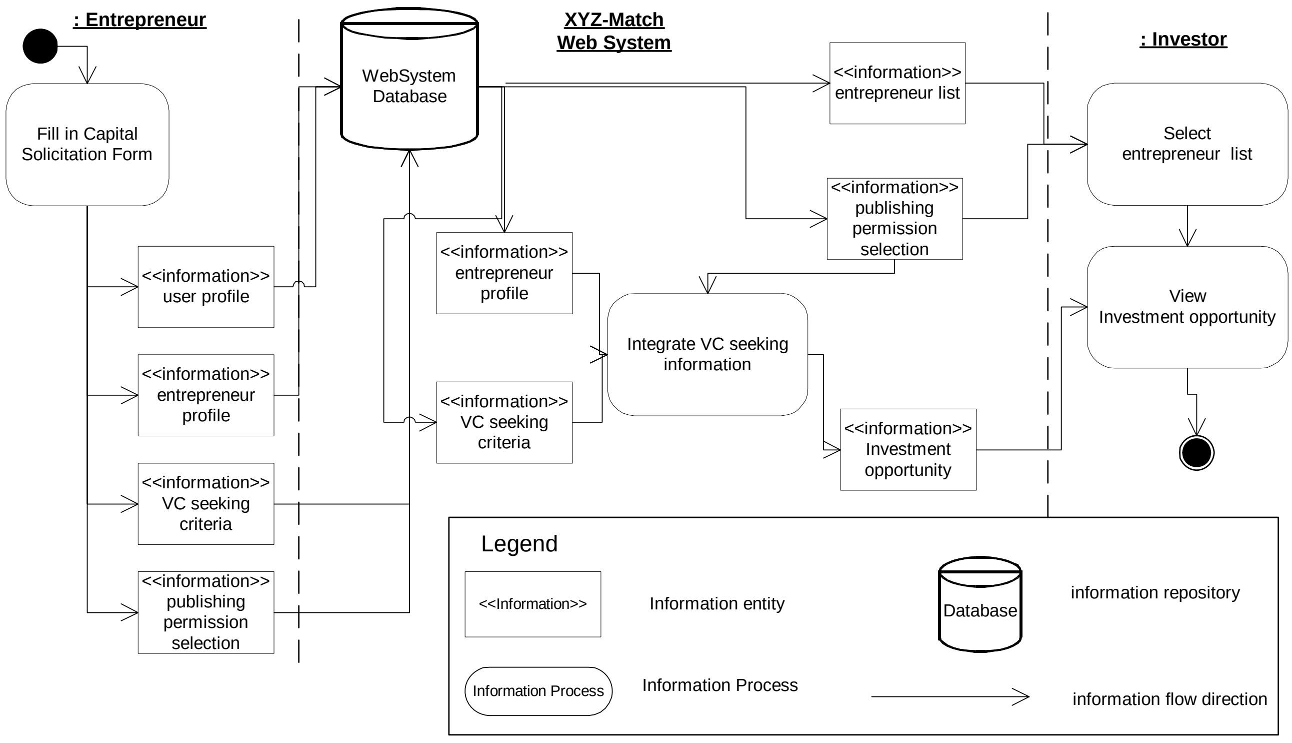


Figure 17. Information structure

Information behaviour is modelled as information flow. Information architects derive the user interface flow diagrams into information flow diagrams to model the information creation, exchange, process and consumption that support the user interface flows. Information diagrams include information, information source and destination, information flow direction, and information processing unit. Figure 18 is a part of an information flow diagram of the capital solicitation scenario derived from the user interface flow in the prototyping behaviour model.



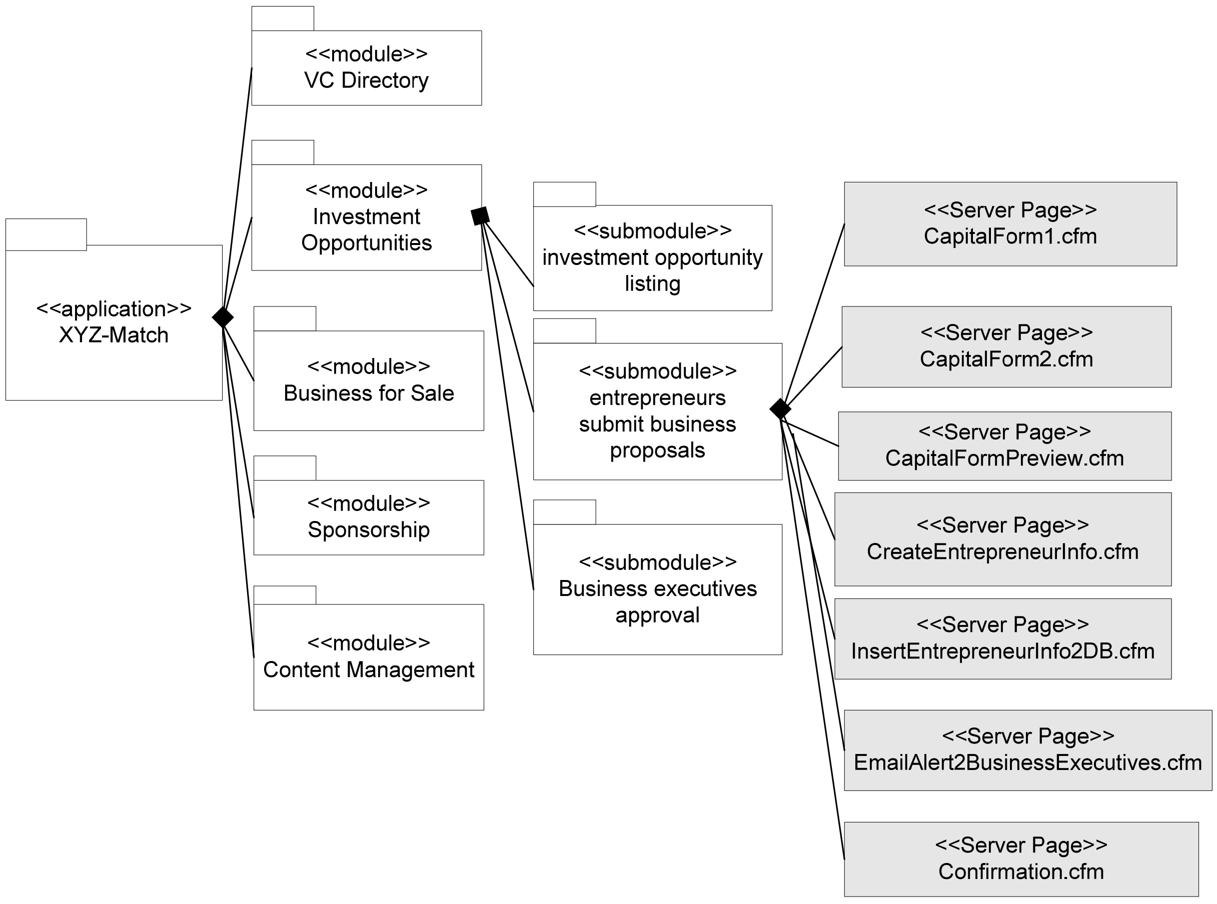
: Entrepreneur
XYZ-Match Web System
: Investor
Fill in Capital Solicitation Form
<<information>> user profile
<<information>> entrepreneur profile
<<information>> VC seeking criteria
<<information>> publishing permission selection
WebSystem Database
<<information>> entrepreneur profile
<<information>> VC seeking criteria
<<information>> entrepreneur list
<<information>> publishing permission selection
Integrate VC seeking information
<<information>> Investment opportunity
Select entrepreneur list
View Investment opportunity
Legend
<<Information>>
Information entity
Database
information repository
Information Process
Information Process
information flow direction

Figure 18. Information behaviour: information flow diagram

5.2.3 System Architecting Step

Once the information model is finished, the system architects begin “System Architecting” to construct the web application design to modules and break down the modules into source code. Modelling *System Architecture Structure* and *System Architecture Behaviour* are the major activities in this step.
In *System Architecture Structure*, system architects cluster the user interfaces into modules. If the modules are complex, break down the modules into submodules. Within each module and submodule, source code (or server pages) is included to realize the user interfaces in *User Interface Structure* and to generate the information (or content) in the *Information Structure*. *System Architecture Behaviour* specifies workflow or business logic within the modules and submodules. It realizes each process in the information flow in the *information behaviour* through source code flow (or server page flow). Each source code flow generates the user interface flow in the *user interface behaviour* and the information flow in the *information behaviour*.

In the system architecting of XYZ-Match application, the modules are clustered by user groups: investors, entrepreneurs, business executives and site administrator. User groups are identified in analysis step. The *system architecture structure* is as Figure 19. For example, the module “Investment opportunities” is broken down to submodules “investment opportunity listing”, “entrepreneurs submit business proposals” and “Business executives’ approval”. The submodule “entrepreneurs submit business proposals” includes a set of source code to generate the user interfaces and information that modelled in the prototyping and information modelling steps. The flow of this set of source code is modelled in the *system architecture behaviour* to realise the user interface flow and the information flow for “entrepreneurs submit business proposals” scenario. The source code flow diagram is as Figure 20. Detailed algorithms within each source code are not considered in the system architecting step, they will be developed in the implementation step by coders.



<<module>> VC Directory
<<module>> Investment Opportunities
<<application>> XYZ-Match
<<module>> Business for Sale
<<module>> Sponsorship
<<module>> Content Management
<<submodule>> investment opportunity listing
<<submodule>> entrepreneurs submit business proposals
<<submodule>> Business executives approval
<<Server Page>> CapitalForm1.cfm
<<Server Page>> CapitalForm2.cfm
<<Server Page>> CapitalFormPreview.cfm
<<Server Page>> CreateEntrepreneurInfo.cfm
<<Server Page>> InsertEntrepreneurInfo2DB.cfm
<<Server Page>> EmailAlert2BusinessExecutives.cfm
<<Server Page>> Confirmation.cfm

Figure 19. System architecture structure

The artifacts of the design steps, “user interface structure”; “user interface behaviour”; “information structure”; “information behaviour”; “system architecture structure” and “system architecture behaviour” will be transformed into the implementation and deployment step.

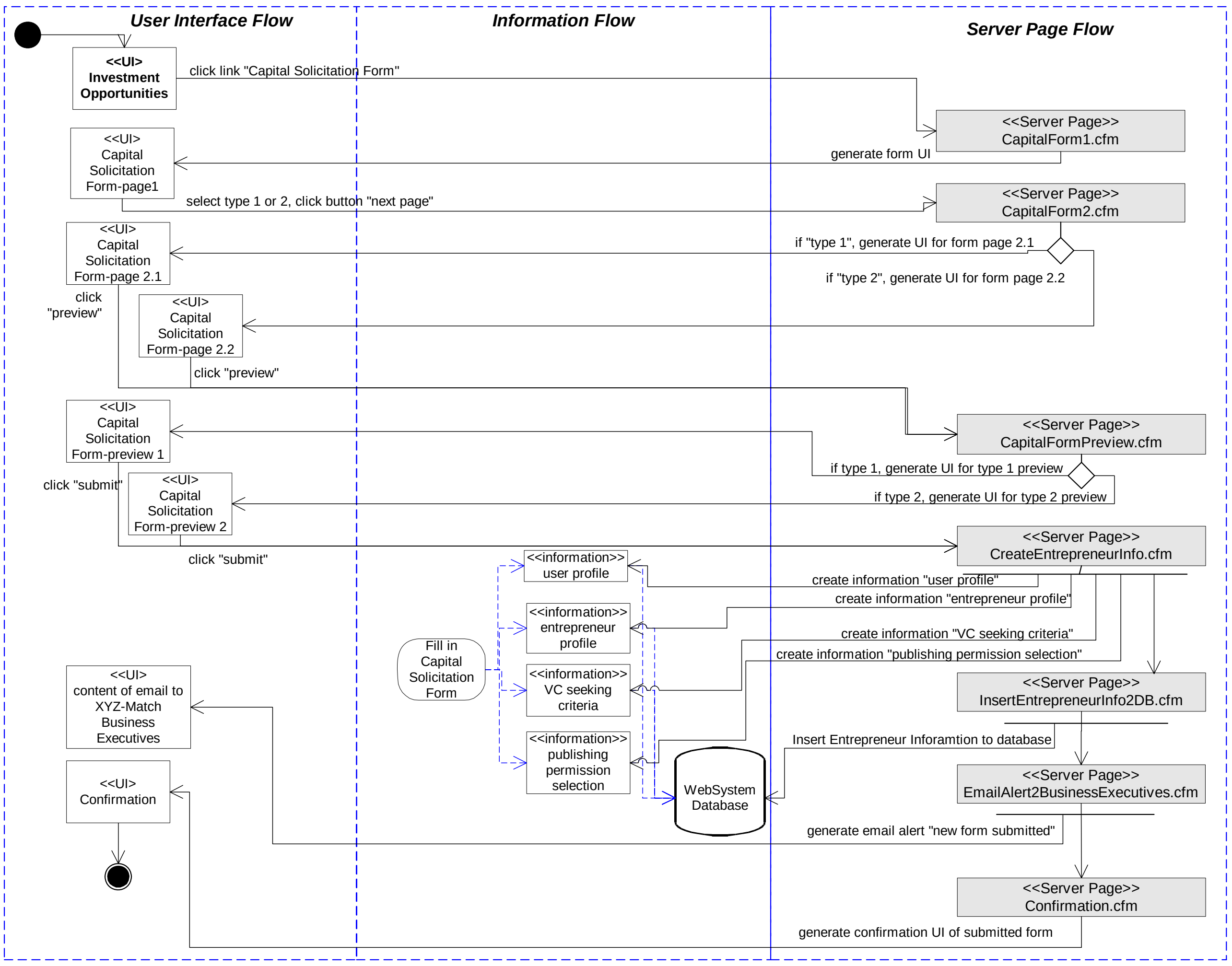

User Interface Flow
Information Flow
Server Page Flow
<<UI>> Investment Opportunities
click link "Capital Solicitation Form"
<<Server Page>> CapitalForm1.cfm
generate form UI
<<UI>> Capital Solicitation Form-page1
select type 1 or 2, click button "next page"
<<Server Page>> CapitalForm2.cfm
<<UI>> Capital Solicitation Form-page 2.1
if "type 1", generate UI for form page 2.1
if "type 2", generate UI for form page 2.2
click "preview"
<<UI>> Capital Solicitation Form-page 2.2
click "preview"
<<UI>> Capital Solicitation Form-preview 1
<<Server Page>> CapitalFormPreview.cfm
if type 1, generate UI for type 1 preview
if type 2, generate UI for type 2 preview
click "submit"
<<UI>> Capital Solicitation Form-preview 2
click "submit"
<<Server Page>> CreateEntrepreneurInfo.cfm
<<information>> user profile
create information "user profile"
<<information>> entrepreneur profile
create information "entrepreneur profile"
create information "VC seeking criteria"
create information "publishing permission selection"
Fill in Capital Solicitation Form
<<information>> VC seeking criteria
<<UI>> content of email to XYZ-Match Business Executives
<<Server Page>> InsertEntrepreneurInfo2DB.cfm
<<information>> publishing permission selection
Insert Entrepreneur Inforamtion to database
<<UI>> Confirmation
WebSystem Database
<<Server Page>> EmailAlert2BusinessExecutives.cfm
generate email alert "new form submitted"
<<Server Page>> Confirmation.cfm
generate confirmation UI of submitted form

Figure 20. Server page flow diagram

6. Conclusions

In order to address the specific characteristics of web-centric systems, this paper presents a trace model for web systems based on Web Application Architecture Framework (WAAF). The trace model has two dimensions: within viewpoint; and amongst viewpoints. Within each viewpoint the artifacts that belong to a stakeholder in the web system life cycle are traced between structure models and behaviour models. Between each viewpoint the needs of different stakeholders are transmitted into other viewpoints by tracing relationships among structures and behaviours.

Traceability is established through linking different artifacts of a development process. The focus of the trace model presented in this paper is on identifying these links. The strength of tracing which provides more accurate, quantitative descriptions of traceability needs to be investigated in future studies.

The use of this trace model is demonstrated through its application in combination with the WAFF-Design process to a commercial web project. The WAAF-Design method focuses on 3 steps: prototyping, information modelling and system architecting. The design process commences from prototyping. Back-end design activities such as “information modelling” and “system architecting” support the front-end prototyping. Information modelling is made explicit by modelling information structure and information flow. The design method guides the activities through the artifacts modelling and transformation at each step. To cope with the system complexity, design activities at each step are partitioned into structural design and behavioural design.

References

Ambler, S. W. (2004). The Object Primer. 3rd Edition, Agile Model Driven Development with UML 2, Cambridge University Press.

ATL http://www.eclipse.org/gmt/atl/ . Last accessed: 9 Nov. 2006

Balmas, F. (2004.). "Displaying dependence graphs: a hierarchical approach." Journal of software maintenance and evolution: research and practices **16**(3): 151-185.

Burdman, J. (1999). Collaborative Web Development: Strategies and Best Practices for Web Teams. Boston, Addison-Wesley.

Cernosek G., N. E. (2004). "The value of modeling." The Rational Edge **Nov**. 2004

Chen, K., & Rajlich, V. (2000). Case Study of Feature Location Using Dependence Graph. International Workshop on Program Comprehension.

Domges, R., Pohl, K., & Schreck, K. (1998). A Filter-Mechanism for method-Driven Trace Capture. The 10th International Conference on Advanced Information System Engineering, Pisa, Italy.

Eclipse Foundation (2006). ALT Home page, http://www.eclipse.org/gmt/atl/ Last accessed: 11 Nov. 2006

Fusebox Lifecycle Process (2005). http://www.fusebox.org/. Last accessed: 9 Nov. 2006

Gotel, O. (1996). Contribution structures. London, England, Imperial College of Science, Technology, and Medicine.

Haumer, P., Pohl, K., & Weidenhaupt, K.. (1998). "Requirements elicitation and validation with real world scenes." IEEE Transactions on Software Engineering **24**(12): 1036-1054.

IBM (2005). IBM Rational Software, http://www-306.ibm.com/software/rational/. Last accessed: 8 July 2005.

IEEE (1990). IEEE Standard Glossary of Software Engineering Terminology. Std 610.12-1990.

IEEE (2000). IEEE Recommended practice for architectural description of software-intensive systems. Std 1471-2000

Isakowitz, T., Stohr, E., & Balasubramanian, P. (1995). "RMM: A Methodology for Structured Hypermedia Design." Communications of the ACM **38**: 34-44.

ISO/IEC (1996a). "Information technology -- Open Distributed Processing -- Reference Model: Foundations -Part 2", ISO/IEC 10746-2:1996

ISO/IEC (1996b). "Information technology -- Open Distributed Processing -- Reference Model: Architecture - Part 3", ISO/IEC 10746-3

ISO/IEC (1998a). "Information technology -- Open Distributed Processing -- Reference model: Overview Part 1", ISO/IEC 10746-1:1998

ISO/IEC (1998b). Information technology -- Open Distributed Processing -- Reference Model: Architectural semantics - Part 4, ISO/IEC 10746-4:1998

Kong, X., Liu, L., & Lowe, D. (2005). "Separation of concerns, a web application architecture framework." Journal of digital information. Volume 6, Issue 2.

Koskinen, J., Salminen, A., & Paakki, J. (2004). "Hypertext support for the information needs of software maintainers." Journal of software maintenance and evolution: research and practices **16**(3): 187-215.

Lange, D. (1994). An Object-Oriented Design Method for Hypermedia Information Systems. Proceedings of the Twenty Seventh Hawaii International Conference on System Sciences, Maui, Hawaii.

Lee, S. C. (1997). A Structured Navigation Design Method For Intranets. Presented at Third Americas Conference on Information Systems, Association for Information Systems (AIS), Indianapolis.

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

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

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.

Murugesan, S. (2000). Web Engineering for Successful Web Application Development. Asia Pacific Web Conference, AeIMS Research Centre, Xian, China.

Object Management Group (2005a), MOF QVT Final Adopted Specification, Object Management Group, OMG doc. Ptc/05-11-01

Object Management Group (2005b). UML: http://www.uml.org/. Last accessed: 9 Nov. 2006

O'Neal, J. S., & Carver, D.L. (2001). Analyzing the impact of changing requirements. IEEE Int. Conference on Software Maintenance.

Palmer, J. D. (1996.). Traceability. Software Requirements Engineering. T. M. Dorfman, R., IEEE Computer Society**:** 266-276.

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

Platt, M. (2002). Microsoft Architecture Overview. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnea/html/eaarchover.asp?frame=true&hidetoc=true, Last accessed: 21 June, 2004.

Podgurski, A., & Clarke, L. (1990). "A formal model of program dependences and its implications for software testing, debugging and maintenance." IEEE Transactions on Software Engineering **16**(9): 965-979.

Pohl, K., Weidenhaupt, K., Domeges, R, Haumer, P., Jarke, M., & Klamma, R. (1999). "PRIME - Toward Process-integrated Modeling Environments." ACM Transactions on Software Engineering and Methodology **8**(4): 343-410.

Powell, T. A., Jones,D.L., et al. (1998). Web Site Engineering: Beyond Web Page Design, Prentice Hall.

QVT – Wikipedia http://en.wikipedia.org/wiki/QVT, Last accessed: 11 Nov. 2006

Ramesh, B., & Jarke, M. (2001). "Toward Reference Models for Requirements Traceability." IEEE Transactions on Software Engineering **27**(1): 58 - 93.

Schwabe, R., G. (1998). Developing Hypermedia Applications using OOHDM. Workshop on Hypermedia Development Processes, Methods and Models (Hypertext'98), Pittsburgh, USA.

Sommerville, J. (2001). Software Engineering, Addison Wesley.

Takahashi K., & L., E. (1997). Analysis and Design of Web-based Information Systems. Presented at 7th International WWW Conference, Brisbane, Australia.

Telelogic (2005). Telelogic DOORS, http://www.telelogic.com/. Last accessed: 9 Nov. 2006

Troyer, O. D. & L., C. (1997). WSDM: A user-centered design method for Web sites. Presented at 7th International WWW Web Conference, Brisbane, Australia.

Vitech Corporation (2005). CORE, http://www.vitechcorp.com/. Last accessed: 8 Feb. 2006

von Knethen, A. (2002). Change-Oriented Requirements Traceability. Support for Evolution of Embedded Systems. Proceedings of International Conference on Software Maintenance.

Yu, E. (1993). Modeling Organizations for Information Systems Requirements Engineering. 1st IEEE Symposium on Requirements Engineering, San Diego.

Zachman, J. A. (1987). "A Framework for Information Systems Architecture." IBM Systems Journal **26,** (3).