Visualized Architecture Knowledge Management Collaboration Services

Ashish Kaul
San Jose State University

Visualized Architecture Knowledge Management Collaboration Services

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By, Ashish Kaul
MSCS, CSU – San Jose State University
Spring 2009

ABSTRACT

Software (system) architecture knowledge is a critical element in making effective design/ implementation decisions for Information Technology departments within companies. This knowledge can be codified and/ or personalized so as to harness the advantages and avoid the missed steps of implementers before us. In research of architecture knowledge enablement, there have been a few ventures, including but not limited to, Processcentric Architecture Knowledge Management Environment (PAKME) [3] and Architecture Design Decision Support System (ADDSS) [4]. In study of these ventures, we find modest attempts at focusing on dissecting types of architecture knowledge and enabling access to details through web tools. The purpose of this paper is to document the design and features of a web tool, namely Visualized Architecture Knowledge Management Collaboration Services (VAKMCS) and its approach in providing an innovative way at accessing and interacting with architecture information to make sound investment decision on IT projects.

ACKNOWLEDGEMENTS

Thanks Dr. Teng Moh, for your patience and understanding over the past years while I attempted to do the impossible. Dr. Moh has motivated me to push for excellence in every facet of my life. His constant support and accommodation of personal and professional conflicts against project time lines and deliverables has helped alleviate much of the anxiety associated with completing this project. Thanks much.

I also appreciate the time that both Dr. Agustin Araya and Dr. Sami Khuri have dedicated to participate as project committee members.

It has been a challenging experience completing the requirements for my degree, yet I feel honored to have finished it under the tutelage of the faculty at San Jose State University.

Thank you.

TABLE OF CONTENTS

1.0 INTRODUCTION..9
1.1 KNOWLEDGE MANAGEMENT 9
1.2 ARCHITECTURE PROCESS 11
1.3 PROBLEM ADDRESSED 12
2.0 RELATED WORK 15
2.1 PAKME...15
2.2 ADDSS..18
3.0 VAKMCS 21
3.1 MOTIVATION 22
3.2 SEARCH 24
3.3 TEXTUAL TAGGING 25
3.4 IMAGE TAGGING...28
3.5 VISUALIZATION 30
3.6 FEEDBACK..31
3.7 USER REPUTATION...31
3.8 VERSIONS 32
4.0 DESIGN 34
4.1 DATABASE SCHEMA...36
4.2 DATA ACCESS LAYER 40
4.3 BUSINESS LOGIC & PRESENTATION LAYER...41
4.4 VAKMS SOFTWARE DESIGN 42
4.4.1 TAG CLOUDS..43
4.4.2 SEARCH RESULTS..44
4.4.3 BROWSE RESULTS...44
4.4.4 IMAGE MAPPING..45
4.4.5 USER REPUTATION..47
4.4.6 VISUALIZATION 48
4.4.7 VERSIONS 49
5.0 SOFTWARE TOOLS, DEVELOPMENT KITS USED 51
6.0 VAKMCS VS. PAKME 54
6.1 SOFTWARE COMPONENTS 54
6.2 KNOWLEDGE DEFINITION...54
6.3 FEATURE COMPARISON...55
6.3.1 SEARCH/ BROWSE 56
6.3.2 VERSIONS 56
7.0 VAKMCS VS. ADDSS 57
7.1 SOFTWARE COMPONENTS 57
7.2 KNOWLEDGE DEFINITION...57
7.3 FEATURE COMPARISON...58
7.3.1 BROWSE 59
7.3.2 VERSIONS 59
8.0 POSSIBLE FUTURE WORKS 60
8.1 CHAT & EMAIL 60

8.2 SEARCH ENGINE 60
8.3 VERSIONS 60
8.4 VISUALIZATION 61
8.5 REPORTING 61
8.6 USER & IMAGE PERSPECTIVE 61
9.0 POSSIBLE REAL WORLD IMPLEMENTATION 62
9.1 PROCESS..64
9.2 GOVERNANCE 64
10.0 REFERENCES...66

LIST OF TABLES

1. Approaches to Architectural Knowledge Management 13
2. The entities and their definition as supported by VAKMCS...21
3. A breakdown of actions leading to different pages and content in VAKMCS.............22
4. A breakdown of the pages and the tag functions implemented in VAKMCS...............26
5. A table containing the details associated with the data layer of VAKMCS 36
6. A list of common functions across the entities found in the data access layer.............40
7. Basic comparison of VAKMCS and PAKME...55
8. Basic comparison of VAKMCS and ADDSS..58

LIST OF FIGURES

1. Architectural Knowledge Management Strategies in Research and Industry 11
2. A diagrammatic representation of the project lifecycle for Cisco Systems, Inc............12
3. The interface to capture a general scenario within PAKME 15
4. A template to present patterns in PAKME 16
5. A process model of reusing design options for PAKME...16
6. A meta-model for architecture design decisions..19
7. A stacked look at the entities and features of VAKMCS 22
8. VAKMCS home page contains the top ten users of the site..23
9. A screenshot of search results page on VAKMCS 25
10. A screen shot that shows how related tags are displayed with the results...................27
11. A screenshot of the details page showing different tagging functions 27
12. A screenshot of the image upload UI..29
13. A screenshot of the image after it has been tagged...29
14. A screenshot of visualization feature on VAKMCS...31
15. A screenshot of Feedback and User reputation...32
16. A screenshot of the version feature...33
17. Layered technology stack for VAKMCS..34
18. Software architecture of VAKMCS...35
19. An ER diagram of VAKMCS entities and their relationships.....................................38
20. An ER diagram of VAKMCS entities and tags, images & feedback 39
21. A block diagram of the business logic and presentation layer of VAKMCS 42
22. Top ten tags flow 43
23. Search flow 44
24. Browse flow..45
25. Image add (left) and view (right) flow..46
26. View user reputation flow...47
27. Add user rating flow 48
28. Visualization flow..49
29. Versions flow...50
30. A possible VAKMCS integration at Cisco 50

1.0 INTRODUCTION

Organizations like Cisco Systems, Inc. spend millions of dollars on a yearly basis in implementing/ procuring information technology (IT) system solutions to meet the needs of their business/ clients. While each organization has its own means of authorizing IT projects (e.g. Portfolio Management Office), there is a lack of support in making informed decisions regarding investment on a new IT solution or re-using existing solutions. Leveraging architectural and system information regarding current portfolio of applications/ solutions to make decisions on how to meet business needs is the core focus of this paper. It is proposed that by implementing software (system) architecture knowledge base that supports association of community driven context can benefit a company in making informed decisions on where to invest their dollars when it comes to IT solutions.

This section highlights the significance of architecture knowledge management and the context for utilization of this knowledge in an information technology project lifecycle. It describes the benefits of and pitfalls from lack of leveraging architectural knowledge for design and implementation decisions in a project. The specific issues around architecture knowledge addressed in this project are discussed in section 1.3.

1.1 KNOWLEDGE MANAGEMENT

Illustrated by Zack [1999a], who states that business organizations are coming to view knowledge as their most valuable and strategic resource. Nonaka [1998] agrees, saying that “in an economy where the only certainty is uncertainty, the one sure source of lasting competitive advantage is knowledge. [1]

Knowledge management (KM) is a field that continues to evolve in its approach and has led to the branching of specialized applications for intellectual property management within a specific domain. KM has been categorized to have multiple approaches like:

- Codification, aimed at making tacit knowledge explicit and;
- Personalization, intended to support knowledge sharing by describing who knows what.

Codification strategy does make sense when addressing architectural content since architects using the information themselves are quite savvy with information systems and adapt/ adopt quickly. The nature of an architect's vocation requires the codifying information through modeling techniques and identifying architectural patterns. [2] Cost of codification is outweighed by benefits due to high-level reusable representation/ storage of the architectural data. Industry as a whole has focused on personalization methodology but is moving towards codification.

Well articulated potential impacts of improper information systems architectural documentation management is provided [2] as:

- The evolution of a system becomes complex and cumbersome, resulting in violations of the fundamental design decisions.
- Inability to identify design errors.
- Inadequate clarification of arguments and information sharing about the design and process.

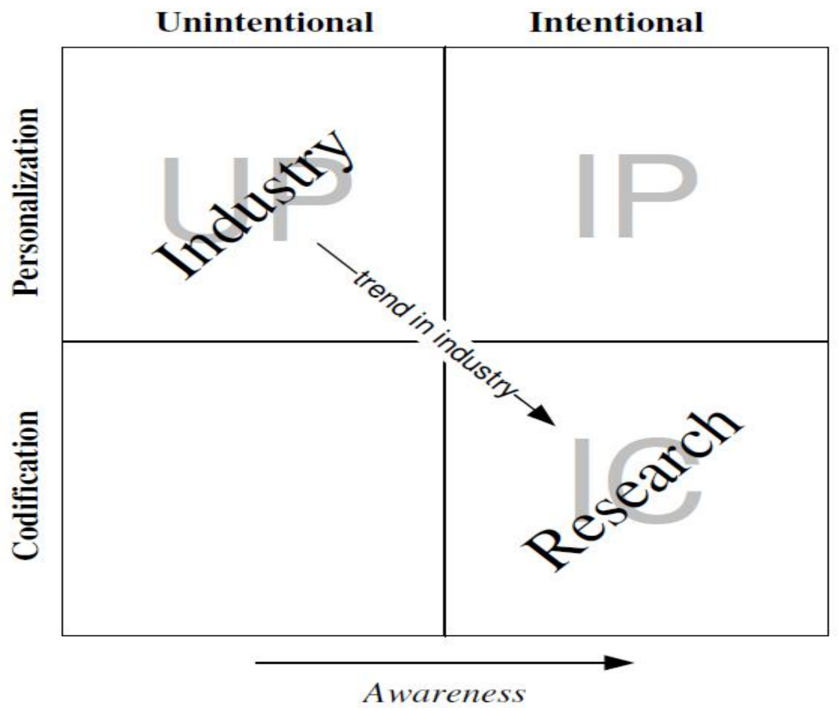


Figure 1. Architectural Knowledge Management Strategies in Research and Industry
Source: [1]

The above figure describes the trend for Architectural Knowledge Management (AKM) and using this as well as one of the suggest approaches by Babar [2]:

- Phase 1: Use personalization for "decision making process is a rather unstructured process in which the architectural solution space is explored and ideas are coined" This way architects will be able to maintain/ leverage knowledge that might be tougher to codify (i.e. stored in a defined manner).
- Phase 2: Codification used for "the design space is outlined by approved architectural decisions, and a stable architectural design emerges."

1.2 ARCHITECTURE PROCESS

Described within this section of the document are details associated with the overall architecture process. This process definition has been defined leveraging the Cisco Systems, Inc. internal project lifecycle documentation.

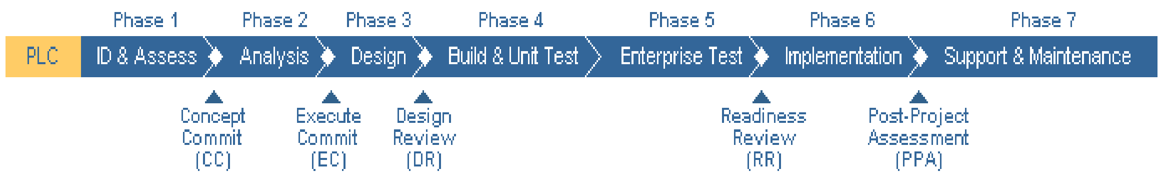


Figure 2. A diagrammatic representation of the project lifecycle for Cisco Systems, Inc.

Essentially within the lifecycle, the major architectural deliverables reside within Phase 1, 2 and 3. Overall, the architect would be consulted throughout the lifecycle; however their major contributions lie in the beginning of the project. The architect is enabled by documents prepared by both business and IT project team members including, but not limited to:

- Business requirements
- Major requirements lists
- Gap analysis [current to future]
- Impact analysis
- Scope

Leveraging these documents the architect then evaluates the current systems and decides on how to best support the needs of the business by accommodating most if not all the requirements by proposing a to-be state (architecture). It is this process that this project will support to improve, by allowing architects to have insight into best practices and already implemented solutions, in turn allowing organizations to invest their IT dollars more wisely.

1.3 PROBLEM ADDRESSED

This project focuses on implementing a solution for architectural knowledge management to allow reusability and validation within an IT organization. Within architectural knowledge management, the specific area of interest is enabling the retrieval of system architecture information for already implemented solutions within a company. To understand the scope and research conducted in this domain, multiple journals and articles were reviewed which confirmed the lack of existing support for appropriate system architecture level knowledge management.

The tool is focused around supporting a process for any company's IT organization to leverage as a "starting point" for all projects. The tool, for now named, Visualized Architectural Knowledge Management Collaboration Services (VAKMCS), will include a subset of features that would finally enable the ultimate goal of managing system architecture information efficiently and effectively. Following below are details of the software implementation for VAKMCS.

As will be highlighted with some details in the related works section of this document, there are potential systems that address the domain of architectural knowledge management, however the implementations are far from a complete solution. The research however provides different approaches and justification for architecture knowledge management. Below is a table that shows some of the known architecture knowledge management systems.

Approach	**Description**
ADDSS	A web-based tool for recording architectural design decisions.
PAKME	A process based knowledge management environment for generic and project-specific knowledge.
DGA DDR	A design decision rationale documentation technique for decision goals and design alternatives.
GRIFFIN	A software architecture project memory to manage know-why and know-how.
RFP	A knowledge repository for reusing best practices with a questionnaire as a front-end.
VCC	Architectural rules disseminated by means of small text-based documents.
RBS	A knowledge base harboring reusable quality criteria.

DSTO	An architectural knowledge management tool to improve architectural evaluation practices.

Table 1. Approaches to Architectural Knowledge Management
Source: [1]

Two specific architecture knowledge management solutions were reviewed in detail and leveraged as guidance for driving definition of this project. In review, it was noted that there is a lack of consolidation of information and collaboration (to enable feedback and design improvements). Both systems have addressed the key factor of tracking design rationale for an architecture, however, the organization of this information is almost segregated from the rest of the knowledge (i.e. to access design rationale, you must choose to explicitly chose to view further details). VAKMCS consolidate and present the architectural information.

There is an intelligent visual factor that is missing from both implementations. VAKMCS will provide features based on image mapping to enhance the search and filtering functionality of the knowledge base.

VAKMCS solution hopes to drive efficiency in using stored knowledge for architectural decision by presenting multiple ways to search. The available options for search are browse by tag or word search.

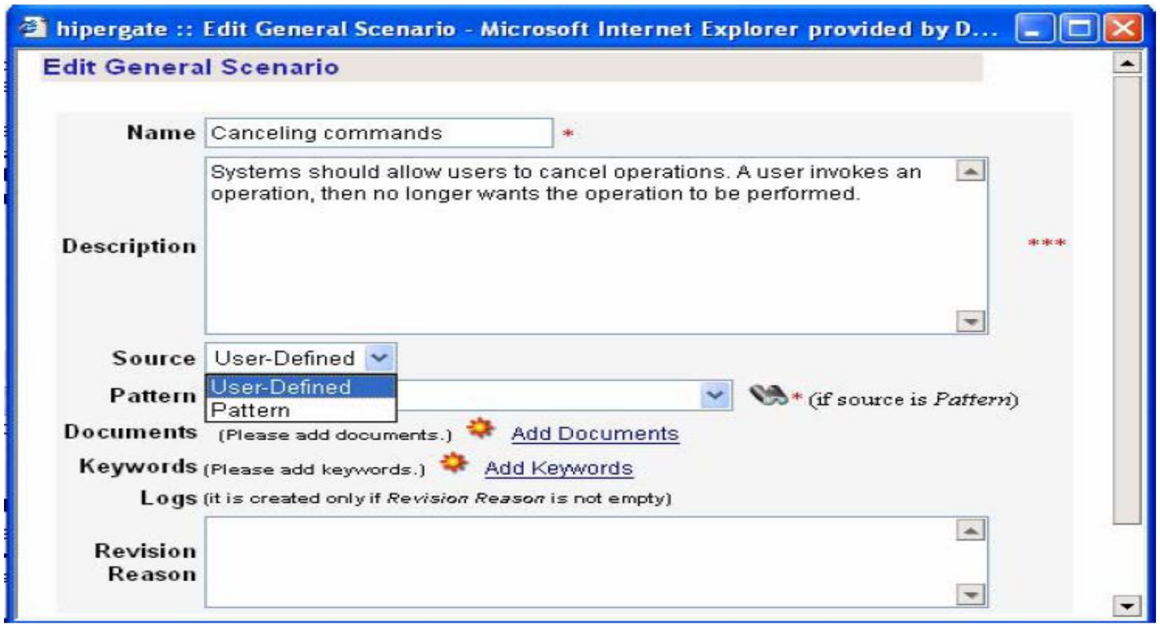
Finally, a major component missing in both implementations is the ability to understand the contributor of architecture. The VAKMCS will enable features that will allow users leverage the reputation of an architecture contributor before deciding on a particular solution for their own project.

2.0 RELATED WORK

This section highlights research conducted in architecture knowledge management domain. Based on [2] details listed in research attempts and implementations at proving an AKM solution, several approaches have been defined. The tools specifically chosen as references for this project for clear definition of implementation were the Processcentric Architecture Knowledge Management Environment (PAKME) [3] and Architecture Design Decision Support System (ADDSS) [4].

2.1 PAKME

The PAKME application has an underlying framework provided by Hipergate, a CRM and groupware open-source project. This tool approaches the AKM problem by providing the following [3]: user interface, knowledge management, repository management, search and reporting. Below are some examples of how PAKME enables such functionality:


hipergate :: Edit General Scenario - Microsoft Internet Explorer provided by D...

Edit General Scenario

Name: Canceling commands *

Description: Systems should allow users to cancel operations. A user invokes an operation, then no longer wants the operation to be performed. ***

Source: User-Defined

Pattern: User-Defined / Pattern * (if source is *Pattern*)

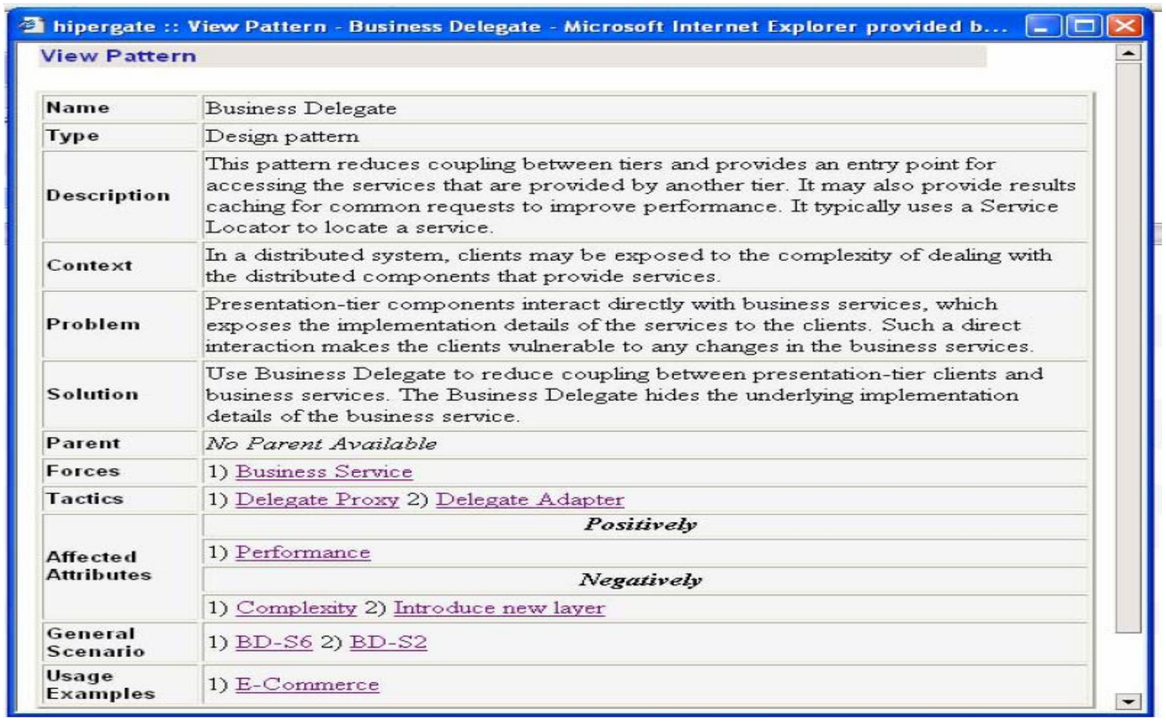
Documents (Please add documents.) Add Documents

Keywords (Please add keywords.) Add Keywords

Logs (it is created only if *Revision Reason* is not empty)

Revision Reason

Figure 3. The interface to capture a general scenario within PAKME
Source: [3]



hipergate :: View Pattern - Business Delegate - Microsoft Internet Explorer provided b...

View Pattern

Name	Business Delegate
Type	Design pattern
Description	This pattern reduces coupling between tiers and provides an entry point for accessing the services that are provided by another tier. It may also provide results caching for common requests to improve performance. It typically uses a Service Locator to locate a service.
Context	In a distributed system, clients may be exposed to the complexity of dealing with the distributed components that provide services.
Problem	Presentation-tier components interact directly with business services, which exposes the implementation details of the services to the clients. Such a direct interaction makes the clients vulnerable to any changes in the business services.
Solution	Use Business Delegate to reduce coupling between presentation-tier clients and business services. The Business Delegate hides the underlying implementation details of the business service.
Parent	*No Parent Available*
Forces	1) Business Service
Tactics	1) Delegate Proxy 2) Delegate Adapter
Affected Attributes	***Positively***
	1) Performance
	Negatively
	1) Complexity 2) Introduce new layer
General Scenario	1) BD-S6 2) BD-S2
Usage Examples	1) E-Commerce

Figure 4. A template to present patterns in PAKME
Source: [3]

The documentation identifies a process that would be used to get the best value out of the tool. This process is defined by the following figure:

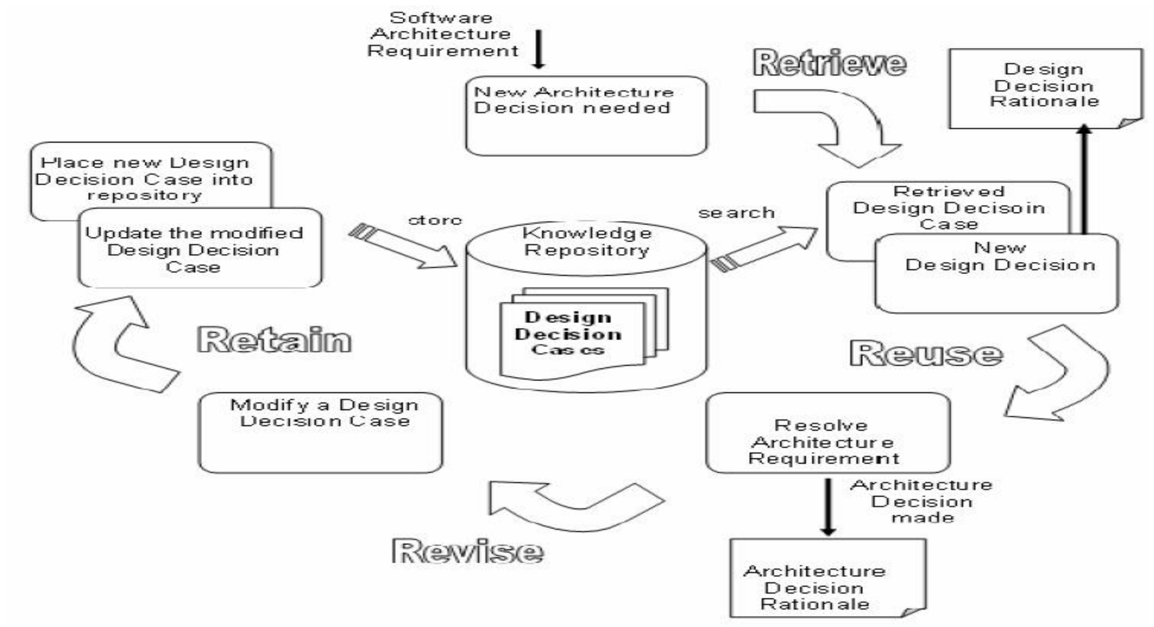

Software Architecture Requirement
New Architecture Decision needed
Retrieve
Design Decision Rationale
Place new Design Decision Case into repository
Update the modified Design Decision Case
store
Knowledge Repository
Design Decision Cases
search
Retrieved Design Decisoin Case
New Design Decision
Retain
Reuse
Modify a Design Decision Case
Resolve Architecture Requirement
Architecture Decision made
Revise
Architecture Decision Rationale

Figure 5. A process model of reusing design options for PAKME
Source: [3]

Strengths of Implementation

This implementation has robust criteria of the knowledge content. The system collects the following data regarding architecture:

General scenario: this is described with a name, a description, a source, date entered and versions.

Pattern: software/ hardware implementation pattern details described with a name, type, description, context, problem, solution, parent (if any), related patterns (if any), forces (if any), tactics (if any), Affected attributes (positive/negative), general scenarios met with this pattern and examples (if any).

Analysis Model: software analysis models described with a name, dependant parameter, independent parameter, equivalent model(s) and rules.

Architecturally Significant Requirement Listing: non-functional requirements described with a name, description, type, analysis model, date proposed, proposed by and quality factor.

Tactic: tactic options for implementation described by name, description, rationale, child of tactic, aim, consequence, strategies, analysis model, applicability, associated rules, assumptions and documents.

Design Option: options described by name, description, notes, rationale, patterns, tactics, constraints, assumptions, rules, documents, consideration for architecture decision, usage in architecture decision and inspiration for other designs.

Glossary: definition of terms.

These criteria drive the ability to well define architecture components. They system considers primarily software development architecture and some system architecture.

The system also enables the knowledge to be stored as either project based or general knowledge. By differentiating between the two, the system allows general knowledge as consideration for a particular project. This delineation of the two could have been used as a means to support collaboration, however, that approach was not addressed.

Weakness of Implementation

The weakness of this implementation is collaboration. The system allows for versioning, however, comments and contributions cannot be addressed without directly changing the details of particular architecture knowledge.

This system also does not enable a physical mapping of content through architecture image capturing. Though you can reference architecture design through documentation, the system does not have the capacity to enable a picture driven interaction with the knowledge.

There are no helpful hints on search, a person has to either browse the entire catalog of information or have strong sense of what they are looking for, and this feature can be counterproductive at times.

2.2 ADDSS

The ADDSS application features include views from multiple perspectives/ user groups, graphical representations, collaboration support, iterative versioning provisions, personalization, software patterns library, designs dependencies and obviously rational for designs. A formal depiction of this feature set is described below with a UML diagram. [4]

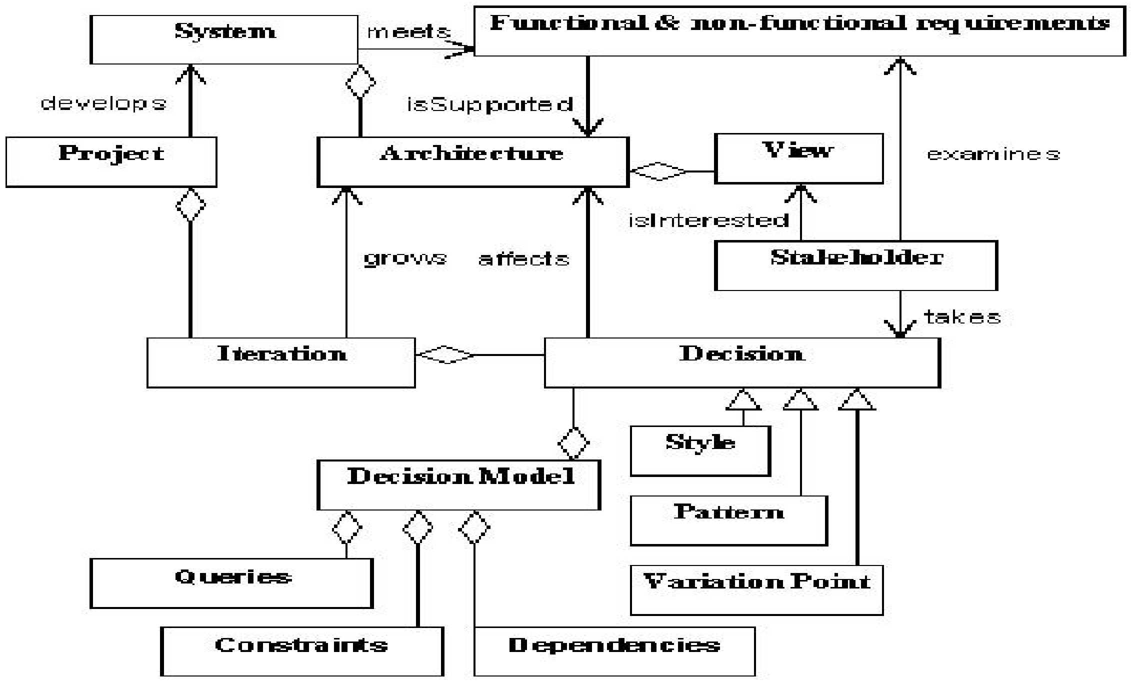


Figure 6. A meta-model for architecture design decisions
Source: [4]

Strengths of Implementation

Important features that are highlighted within the tool include personalization and multiple view perspectives which are potentials for the project.

There is an available library of potential design patterns that can be leveraged for software architecture. The design patterns are described with a name, a description, a type and an image. These can be used to associate with decisions detailed in the system.
Version/ Iteration facility is available but with minimal functionality.

Weakness of Implementation

The review of this tool shows many possible areas of improvement including the usability and lack of intuitive process for storing and retrieving/ using the architectural knowledge. The tool itself is focused on software architecture knowledge, rather than system architecture, but provides a foundation that is applicable for the scope of this project.

Overall the implementation is quite rudimentary and the proposed solution has not yet been realized to its full capacity. There are many disjointed features that need to be cohesive for usability.

3.0 VAKMCS

In this section of the document we will describe the features that are implemented as part of this project. VAKMCS is a web based tool that leverages both existing stored architecture information and enables organic community driven context for better usability. Below is a table that defines the entities for which VAKMCS provides services to access details.

Entity	**Definition**
Requirement	A feature that is enabled by a system or integration.
System	A software application which services a set of requirements.
Integration	Medium of communication between two systems.
Ecosystem	A set of systems and integrations which services business purpose.
Project	An implemented IT project containing details associated with systems, integrations and teams involved.
User	A contributor/ viewer of VAKMCS.

Table 2. The entities and their definition as supported by VAKMCS

In the following sections we will be describing the services provided by VAKMCS to enable informed decision making for IT spending. Below is a stack diagram that portrays the high level solution overview of VAKMCS. The entities essentially depict data sources that are related by the service layer. The services layer not only relates the data sources, but also enables features that facilitate the knowledge access. By allowing entities to be tagged, rated, searched etc. the VAKMCS implementation will attempt to improve the experience of using/ consuming the stored knowledge.

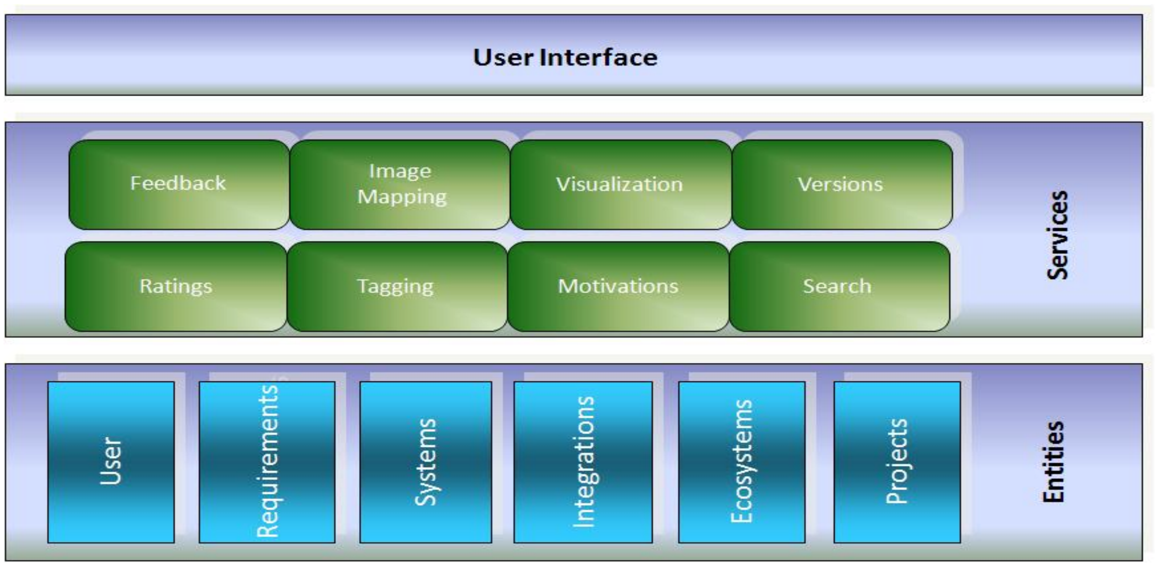


Figure 7. A stacked look at the entities and features of VAKMCS

VAKMCS renders the knowledge on different views based on action a user takes on the system. The actions and the content of resulting pages are described below in the table.

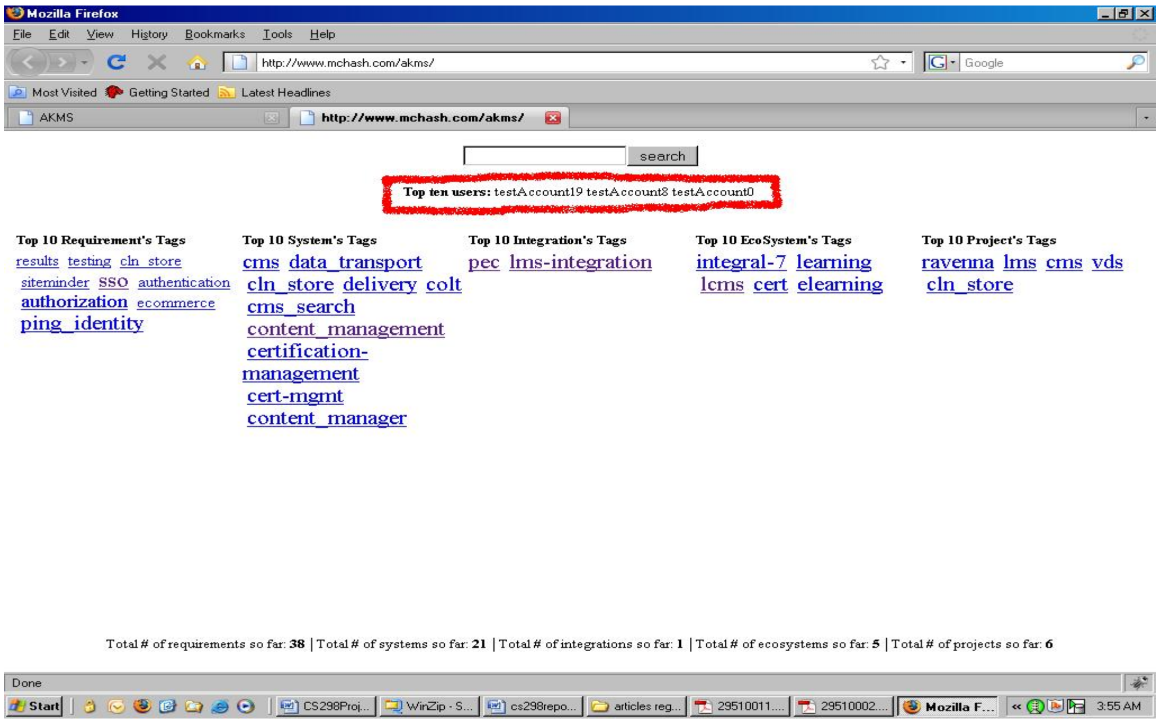
Action	**Page**	**Content**
Access system	Home page	The homepage contains high level activity and knowledge details within the system. The page enables search, browse by tags, and provides statistics on content and users.
Search	Search result page	Renders the results in a grid format matching the search string of a user. Also shows associated tags with the entities matching the search criteria.
Browse by tag	Tag result page	Renders the results in a grid format match the tag clicked by a user. Also shows other associated tags with the resulting entities.
Get details	Details page	Renders all captured details about an entity. Also provides access to other functions like image mapping, visualization, adding tags etc.

Table 3. A breakdown of actions leading to different pages and content in VAKMCS

3.1 MOTIVATION

As is noticed in many companies, more often than not, after any architecture knowledge base is implemented, the community lacks the motivation to update the knowledge and hence discontinues leveraging the system for lack of reliability of information to make appropriate decisions. The solution implemented as part of this project delves into resolving this factor by providing a dynamic look into who has contributed to the site and what rating they receive by their peers. By enabling this feature, it is proposed that there is more likely hood of users to continue to update their knowledge and usage as they may be rewarded for their positive contributions. It is not the intention of this project to provide a complete solution for motivation, but rather a simple implementation that depicts how the system can receive greater adoption and usage.

Below screenshot shows the motivation feature as implemented in VAKMCS. As highlighted (red box) in the screenshot, the top ten users are shown at the entry point of the site. This factor can drive positive competition between peers and improve utilization of the system. The top ten users are discerned based on the highest average rating (out of 5) a contributor of the site receives.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.mchash.com/akms/
Google
Most Visited Getting Started Latest Headlines
AKMS
http://www.mchash.com/akms/
search
Done
Start
CS298Proj...
WinZip - S...
cs298repo...
articles reg...
29510011....
29510002....
Mozilla F...
3:55 AM

Top ten users: testAccount19 testAccount8 testAccount0

Top 10 Requirement's Tags
results testing cln_store siteminder SSO authentication authorization ecommerce ping_identity

Top 10 System's Tags
cms data_transport cln_store delivery colt cms_search content_management certification-management cert-mgmt content_manager

Top 10 Integration's Tags
pec lms-integration

Top 10 EcoSystem's Tags
integral-7 learning lcms cert elearning

Top 10 Project's Tags
ravenna lms cms vds cln_store

Total # of requirements so far: **38** | Total # of systems so far: **21** | Total # of integrations so far: **1** | Total # of ecosystems so far: **5** | Total # of projects so far: **6**

Figure 8. VAKMCS home page contains the top ten users of the site

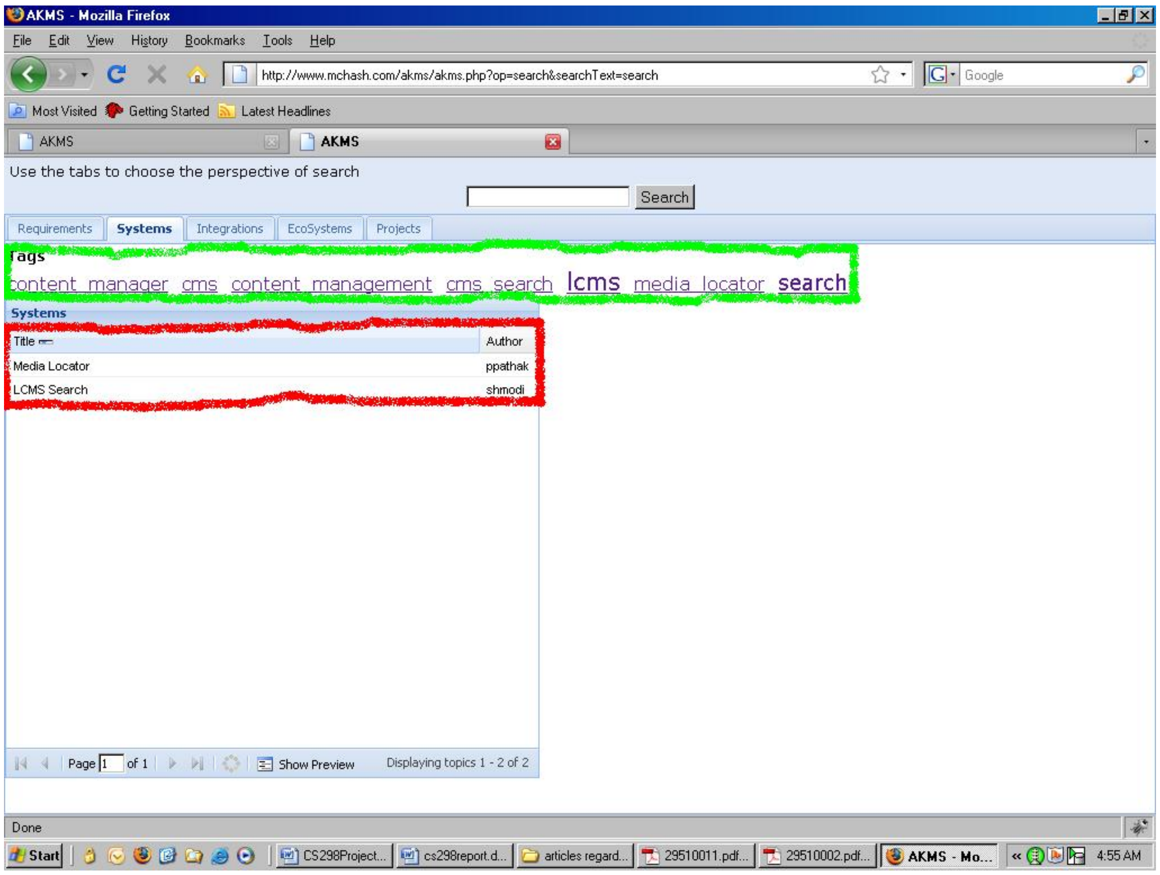
3.2 SEARCH

VAKMCS provides the ability to perform a common search across all architectural entities (requirements, systems, integrations, ecosystems and projects). The search is driven by text matching based on the title and description of an entity. This feature provides the ability for a user to get a holistic view of the architecture information and decide to drill down into the appropriate area (entity) of search based on the initial search results. The initial search results render two assets for a user:

- A title that defines the entity and is clickable for further information
- Tags associated with the entities that match the search criteria

The tags allow for a user to redefine their search criteria to a more appropriate term (as tags are community driven content for organic context definition) or simply use browse by tags to get appropriate results.

Below is a screenshot of VAKMCS search results page. In this screenshot, the user is searching by the text "search" to find systems that provide search services within the company. The results show systems that match the search text (highlighted in red box), as well as, the tags associated with the systems that match the search text (highlighted in green box). The user can choose to change perspective of the search and look within requirements, integrations, ecosystems or projects to see matches for the same criteria.


AKMS - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.mchash.com/akms/akms.php?op=search&searchText=search
Use the tabs to choose the perspective of search
Search
Requirements Systems Integrations EcoSystems Projects
Tags
content manager cms content management cms search lcms media locator search
Systems
Page 1 of 1
Show Preview
Displaying topics 1 - 2 of 2
Done

Title	Author
Media Locator	ppathak
LCMS Search	shmodi

Figure 9. A screenshot of search results page on VAKMCS

3.3 TEXTUAL TAGGING

As depicted in the stack diagram in the introduction of this section, tagging service enables a community driven context to be added to knowledge. Users have the ability to tag entities (requirements, systems, integrations, ecosystems and projects) to provide multiple contexts to the same information.

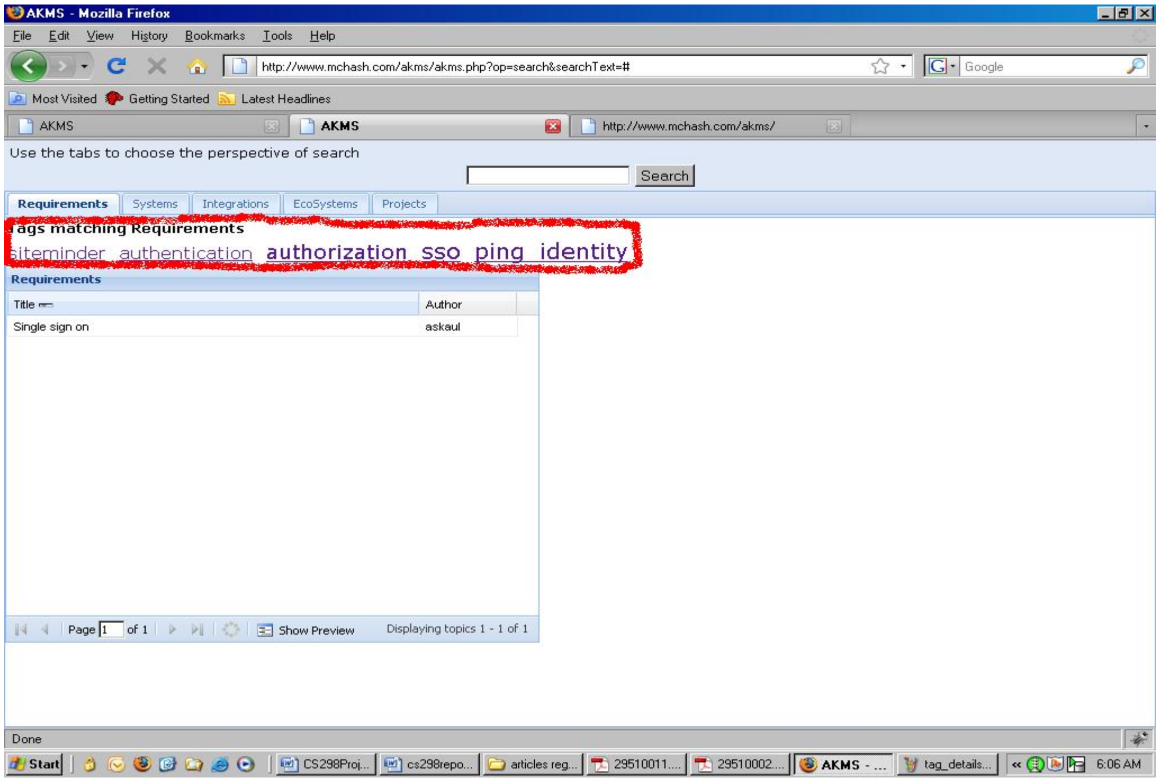
The tag clouds rendered on the web pages are dynamically arrange themselves in varying sizes based on number of times a particular knowledge has been tagged with same text. The more times an item is tagged with the same text, the larger the disparity in size between not as commonly used texts for tags. This feature helps the user understand what are the most common connotations associated with the knowledge they wish to ascertain. By leveraging this feature during browsing the user can also better identify how they should search for the appropriate information within the knowledge base.

Browsing by tags can be leveraged in multiple pages within VAKMCS. Depending on where a user is within the application, there are different options to browse by tags. The pages and the browsing by tag features applied through VAKMCS are described below in the table.

Page	**Tag feature description**
Home page	By providing the top ten tags associated with each entity (requirements, systems, integrations, ecosystems and projects) VAKMCS enables the user to have a starting point for their search. The user can click on the tag and begin their search for the appropriate entity. The top ten tags are determined based on the entity and the maximum number of times a particular knowledge item is tagged with the same text.
Search results page	As described in section 3.2, along with the results of a search by matching the text, the search page provides tags that are associated with the results. Users can choose to browse by tags from search page if they wish.
Tag results page	As in the search page, if a user decides to browse by tag, they will not only be provided by entity that has been associated with the tag, but also other tags associated with the results.
Details page	Within the details page (where all stored details associated with an entity are displayed) a user is provided with all tags associated with all entities associated with given entity. An example of this would be when a user looks at the details of a particular system; they are also shown the related tags for all systems, requirements, ecosystems, integrations and projects that are in some way associated with the system.

Table 4. A breakdown of the pages and the tag functions implemented in VAKMCS

Below are screenshots from VAKMCS tag page and details page that highlight the features described in the above table. On the tag page, we can see all the associated tags associated with entities (shown in red box) that match the tag by which a user is browsing. On the details page we can see the all the associated tags categorized by entities in highlighted in a red box. Also seen on the same page is the ability for a user to add a tag, as seen in the green box and identify a new context to the same information.


AKMS - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.mchash.com/akms/akms.php?op=search&searchText=#
Most Visited Getting Started Latest Headlines
AKMS
http://www.mchash.com/akms/
Search
Requirements Systems Integrations EcoSystems Projects
Page 1 of 1
Show Preview
Done
Start
6:06 AM

Use the tabs to choose the perspective of search

Tags matching Requirements

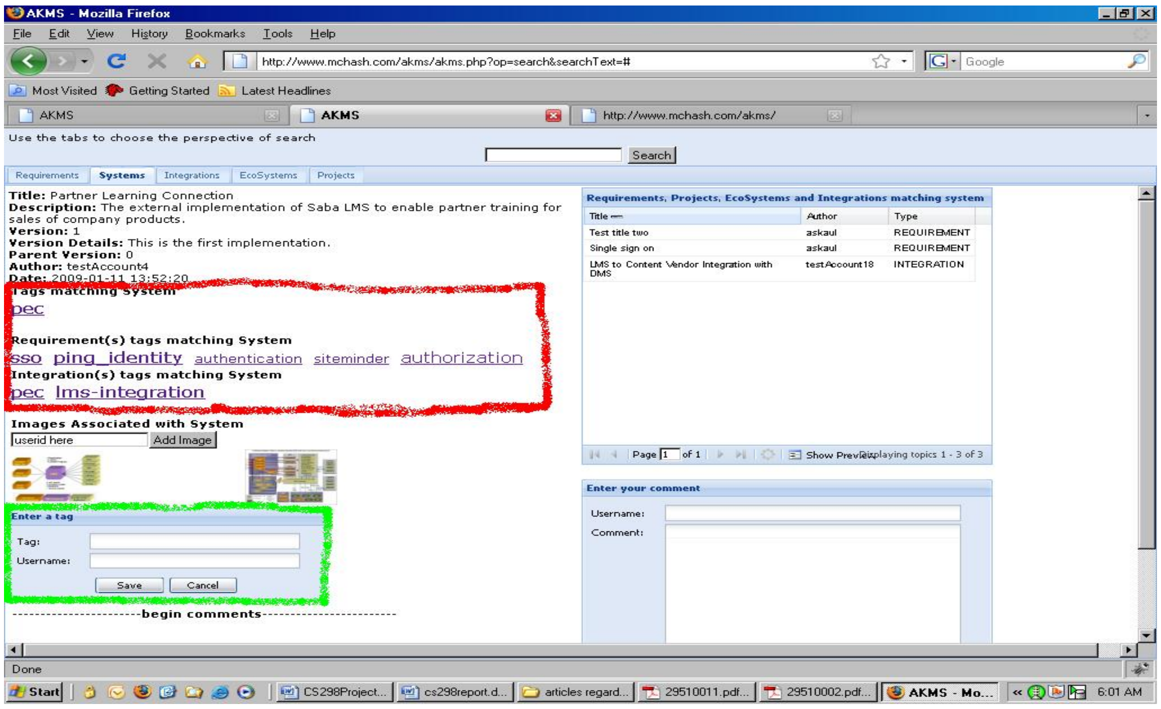
siteminder authentication authorization sso ping identity

Requirements

Title	Author
Single sign on	askaul

Displaying topics 1 - 1 of 1

Figure 10. A screen shot that shows how related tags are displayed with the results


AKMS - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.mchash.com/akms/akms.php?op=search&searchText=#
Most Visited Getting Started Latest Headlines
AKMS
http://www.mchash.com/akms/
Search
Requirements Systems Integrations EcoSystems Projects
Add Image
Save Cancel
Page 1 of 1
Show Preview
Done
Start
6:01 AM

Use the tabs to choose the perspective of search

Title: Partner Learning Connection
Description: The external implementation of Saba LMS to enable partner training for sales of company products.
Version: 1
Version Details: This is the first implementation.
Parent Version: 0
Author: testAccount4
Date: 2009-01-11 13:52:20

Tags matching System

pec

Requirement(s) tags matching System

sso ping_identity authentication siteminder authorization

Integration(s) tags matching System

pec lms-integration

Images Associated with System

userid here

Enter a tag

Tag:
Username:

----------------------**begin comments**-----------------------

Requirements, Projects, EcoSystems and Integrations matching system

Title	Author	Type
Test title two	askaul	REQUIREMENT
Single sign on	askaul	REQUIREMENT
LMS to Content Vendor Integration with DMS	testAccount18	INTEGRATION

Displaying topics 1 - 3 of 3

Enter your comment

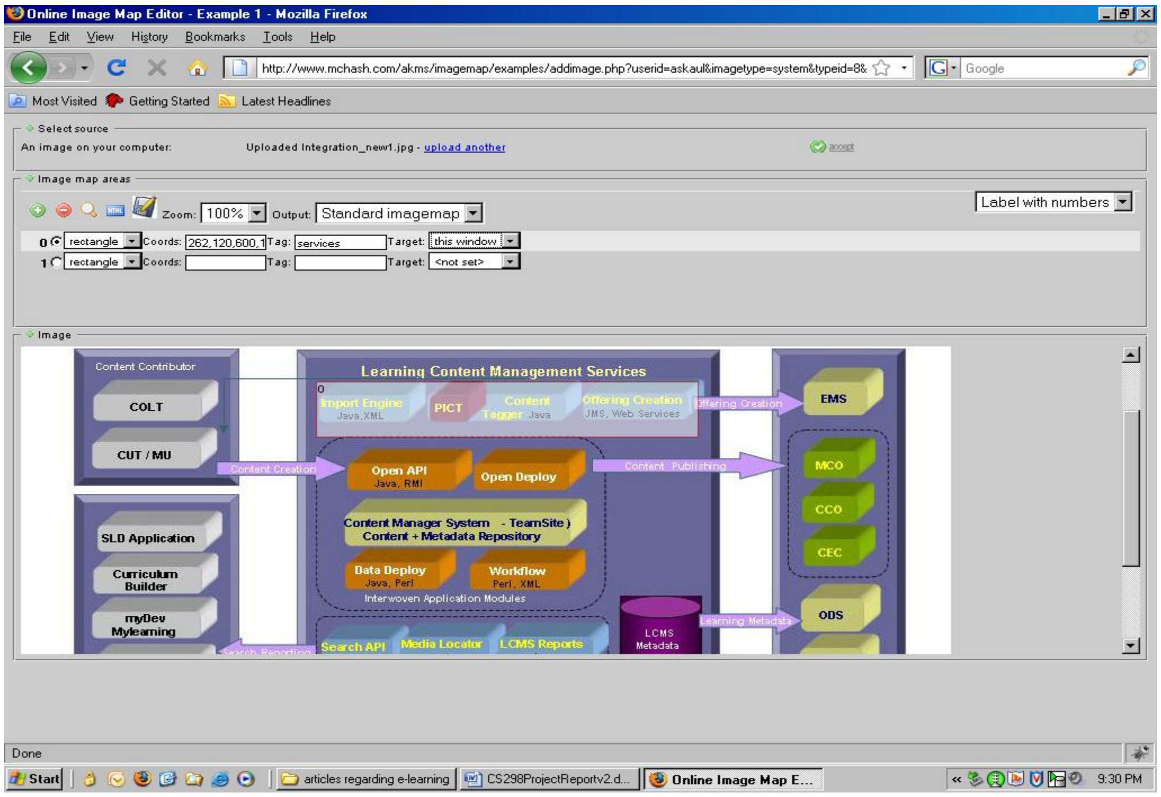
Username:
Comment:

Figure 11. A screenshot of the details page showing different tagging functions

3.4 IMAGE TAGGING

As cliché as it may seem, a picture is worth more than thousand words. With that in mind, VAKMCS, offers the capability to associate images with entities (systems and integrations) via tagging. This feature allows for a user to upload an image and associate tags within certain sections of the picture. Depending on the picture uploaded and how it is tagged, users can get access to details around the ecosystem within which a particular system may reside, its integrations and/ or other significant information. The image tagging capability allows multiple images to be associated with one entity, e.g. an image that depicts the systems categorization within an functional architecture diagram vs. an image that depicts a how a system is integrated with other systems for certain functional requirements.

Below are a couple of screenshots of how a user can leverage this feature within VAKMCS. The first screenshot displays UI a user uses to add an image and tags associated with the image. The second screenshot shows how a user can view the image and entities mapped within the image. In this second screenshot, the user must click on the image as it appears in the details page of an entity to view the tags associated with actual image.


Online Image Map Editor - Example 1 - Mozilla Firefox

http://www.mchash.com/akms/imagemap/examples/addimage.php?userid=askaul&imagetype=system&typeid=8&

Select source

An image on your computer: Uploaded Integration_new1.jpg - upload another

Image map areas

Zoom: 100% Output: Standard imagemap Label with numbers

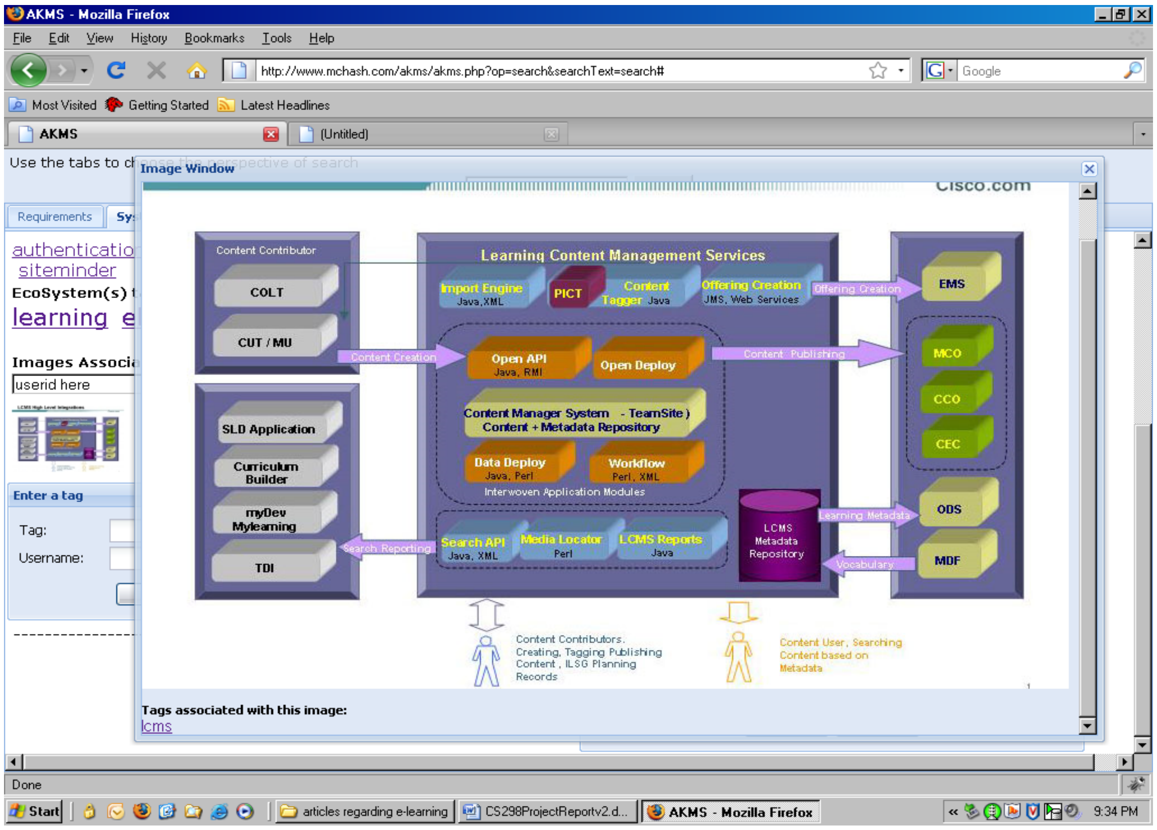
0 rectangle Coords: 262,120,600,1 Tag: services Target: this window

1 rectangle Coords: Tag: Target: <not set>

Image

Done

Figure 12. A screenshot of the image upload UI


AKMS - Mozilla Firefox

http://www.mchash.com/akms/akms.php?op=search&searchText=search#

AKMS (Untitled)

Use the tabs to c

Image Window

Tags associated with this image:
lcms

Done

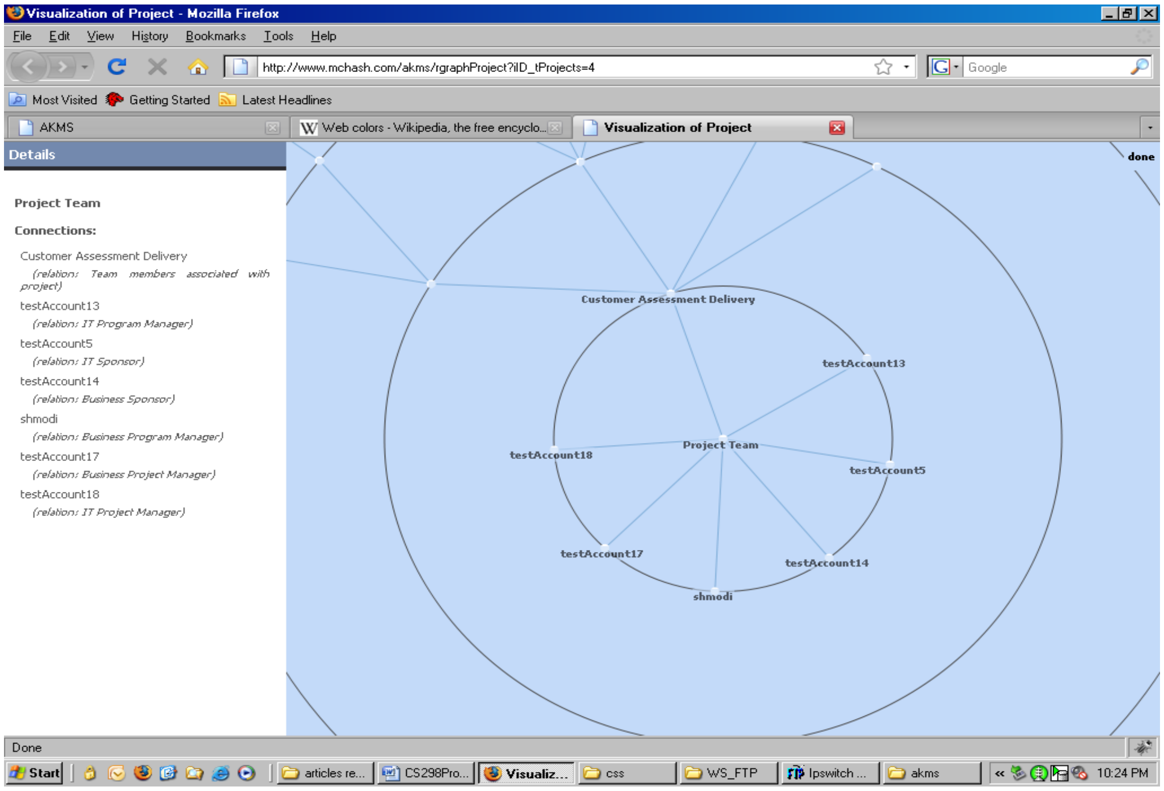
Figure 13. A screenshot of the image after it has been tagged

3.5 VISUALIZATION

Visualization is a feature that enables the user to view the relationships of any entity in a graphical format. This feature provides the user with an interactive graph to view how different entities within the VAKMCS application are related to each other, rather than only a textual display. The utilization of this feature is geared in helping users easily find two critical pieces of information:

- For projects: find the systems and integrations that were delivered as part of the project, as well as, find the project team members and the roles they play. By providing a view into the project team, the user should be go outside of the tool and connect with the appropriate person to get further information about a project that may not be documented anywhere. This is key in the real world where often getting first hand information from an involved party can provide more than just technical information regarding an implementation / project.
- For integrations: find the systems integrated together based on a function. By reviewing this information in a graphical rather than a textual format, a user could get a more helpful view into the data.

Below is a screenshot that shows details associated with a project. The feature provides the details about the relations between elements on the graph in the left panel. As a user navigates the graph, the left panel dynamically updates to depict the relationship details.


Visualization of Project - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.mchash.com/akms/rgraphProject?iID_tProjects=4
Most Visited Getting Started Latest Headlines
AKMS
Web colors - Wikipedia, the free encyclo...
Visualization of Project
Details
done
Customer Assessment Delivery
testAccount13
Project Team
testAccount18
testAccount5
testAccount17
testAccount14
shmodi
Done
Start
articles re...
CS298Pro...
Visualiz...
css
WS_FTP
Ipswitch ...
akms
10:24 PM

Project Team

Connections:

Customer Assessment Delivery
(relation: Team members associated with project)
testAccount13
(relation: IT Program Manager)
testAccount5
(relation: IT Sponsor)
testAccount14
(relation: Business Sponsor)
shmodi
(relation: Business Program Manager)
testAccount17
(relation: Business Project Manager)
testAccount18
(relation: IT Project Manager)

Figure 14. A screenshot of visualization feature on VAKMCS

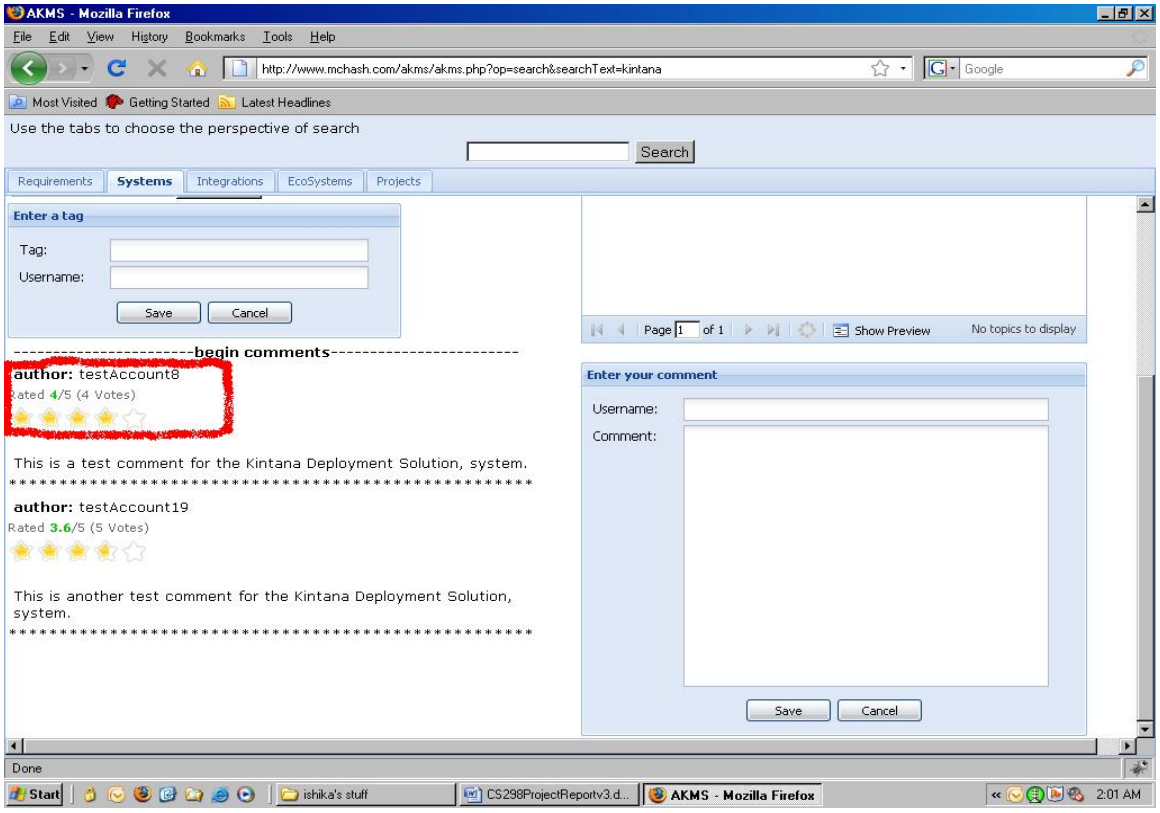
3.6 FEEDBACK

This feature allows the ability for a user to add feedback about a particular entity. This feature focuses on motivating users' to leverage the architecture information and interact with it for validating and/ or questioning the knowledge. Based on the feedback by a user, others can respond and add their thoughts around either a previous comment or the actual entity. Another benefit of this feature is the ability for users to add additional information that may have missed regarding the entity.

3.7 USER REPUTATION

This feature of the application focuses on enabling users to understand the reliability of a user as based by the community. As seen on many web 2.0 sites today, with open access to add content to a site, it's often overwhelming for users to find the appropriate content without getting lost. These features enhance the ability for a user to trust the content based on the rating of the author before leveraging it for their decision. These features are also leveraged in motivating users to interact with the application and get recognition for consistent positive and useful contribution. Star ratings are used to allow users the ability to rate authors and an average score is shown for each author rated.

Below is a screenshot of user reputation feature (shown in red box). Users have the ability to rate by clicking on the stars below the text showing the author of a feedback. The feedback can be viewed directly below the user reputation in the diagram below.


AKMS - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.mchash.com/akms/akms.php?op=search&searchText=kintana
Most Visited Getting Started Latest Headlines
Search
Requirements Systems Integrations EcoSystems Projects
Enter a tag
Tag:
Username:
Save Cancel
Page 1 of 1 Show Preview No topics to display
Enter your comment
Username:
Comment:
Save Cancel
Done
Start ishika's stuff CS298ProjectReportv3.d... AKMS - Mozilla Firefox 2:01 AM

Use the tabs to choose the perspective of search

----------------------begin comments-----------------------
author: testAccount8
Rated 4/5 (4 Votes)

This is a test comment for the Kintana Deployment Solution, system.
****
author: testAccount19
Rated 3.6/5 (5 Votes)

This is another test comment for the Kintana Deployment Solution, system.
****

Figure 15. A screenshot of Feedback and User reputation

3.8 VERSIONS

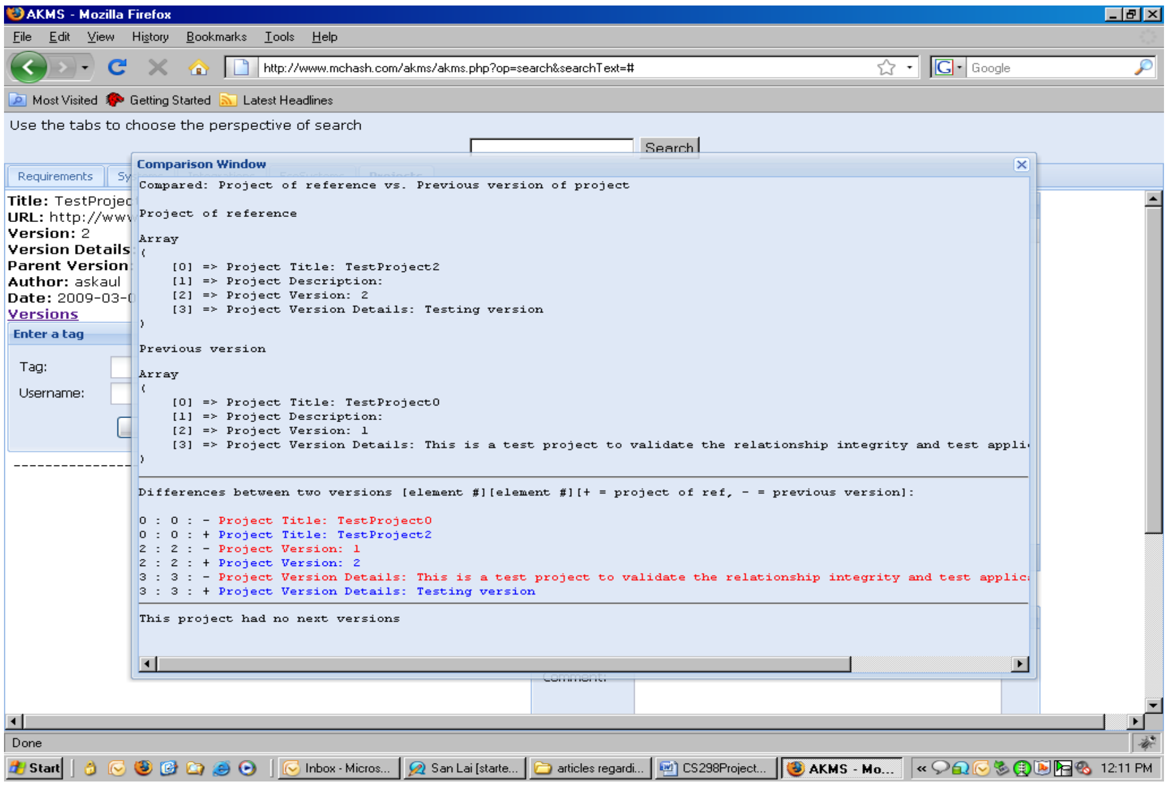
Versions feature on the application supports the ability for a user to do a quick comparison between the details associated with the current entity and its predecessor and successor if available. The feature does a comparison across the following details:

- Entity title

- Entity description
- Entity version
- Entity version description
- Tags associated with the different versions
- Entities associated with the entity (e.g. requirements associated with a system)

By reviewing this information, the user can get an understanding of the past roadmap of the entity and see how well it aligns with a new prospective project that requires IT funding. If details show an alignment, it would be beneficial for the user to reach out to the responsible party and work on a possible avenue to leverage existing platforms.

Below is a screenshot of a project versioning output. The output does a string comparison between the two entities to show differences between current and pre/ post versions.


```
Compared: Project of reference vs. Previous version of project

Project of reference

Array
(
    [0] => Project Title: TestProject2
    [1] => Project Description:
    [2] => Project Version: 2
    [3] => Project Version Details: Testing version
)

Previous version

Array
(
    [0] => Project Title: TestProject0
    [1] => Project Description:
    [2] => Project Version: 1
    [3] => Project Version Details: This is a test project to validate the relationship integrity and test appli
)

Differences between two versions [element #][element #][+ = project of ref, - = previous version]:

0 : 0 : - Project Title: TestProject0
0 : 0 : + Project Title: TestProject2
2 : 2 : - Project Version: 1
2 : 2 : + Project Version: 2
3 : 3 : - Project Version Details: This is a test project to validate the relationship integrity and test applic
3 : 3 : + Project Version Details: Testing version

This project had no next versions
```

Figure 16. A screenshot of the Version feature

4.0 DESIGN

In this section of the document are details regarding the design of VAKMCS. The application was developed using Javascript/ HTML on client side, with PHP on the server side and MySQL as the backend database. The application has a presentation layer preparing the client side code supported by a business logic layer which performs activities like processing data from the database that is provided by the data layer. Below is a diagram that portrays the technology stack and where it resides in the application layer.

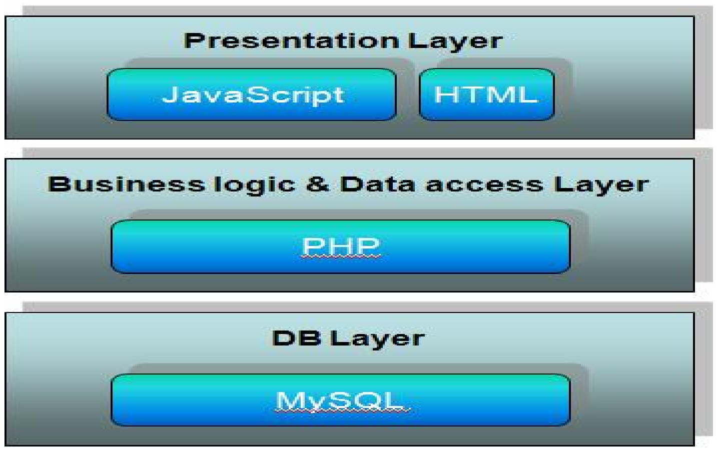


Figure 17. Layered technology stack for VAKMCS

The overall software architecture for VAKCMS leverages multiple software components in aiding the implementation of the feature and functions realized by the application. Below is a software architecture diagram that depicts the software components and functions within the layers of the application, as mentioned above. In the following sections the specifics around the different layers within the application are described with the functions that they perform.

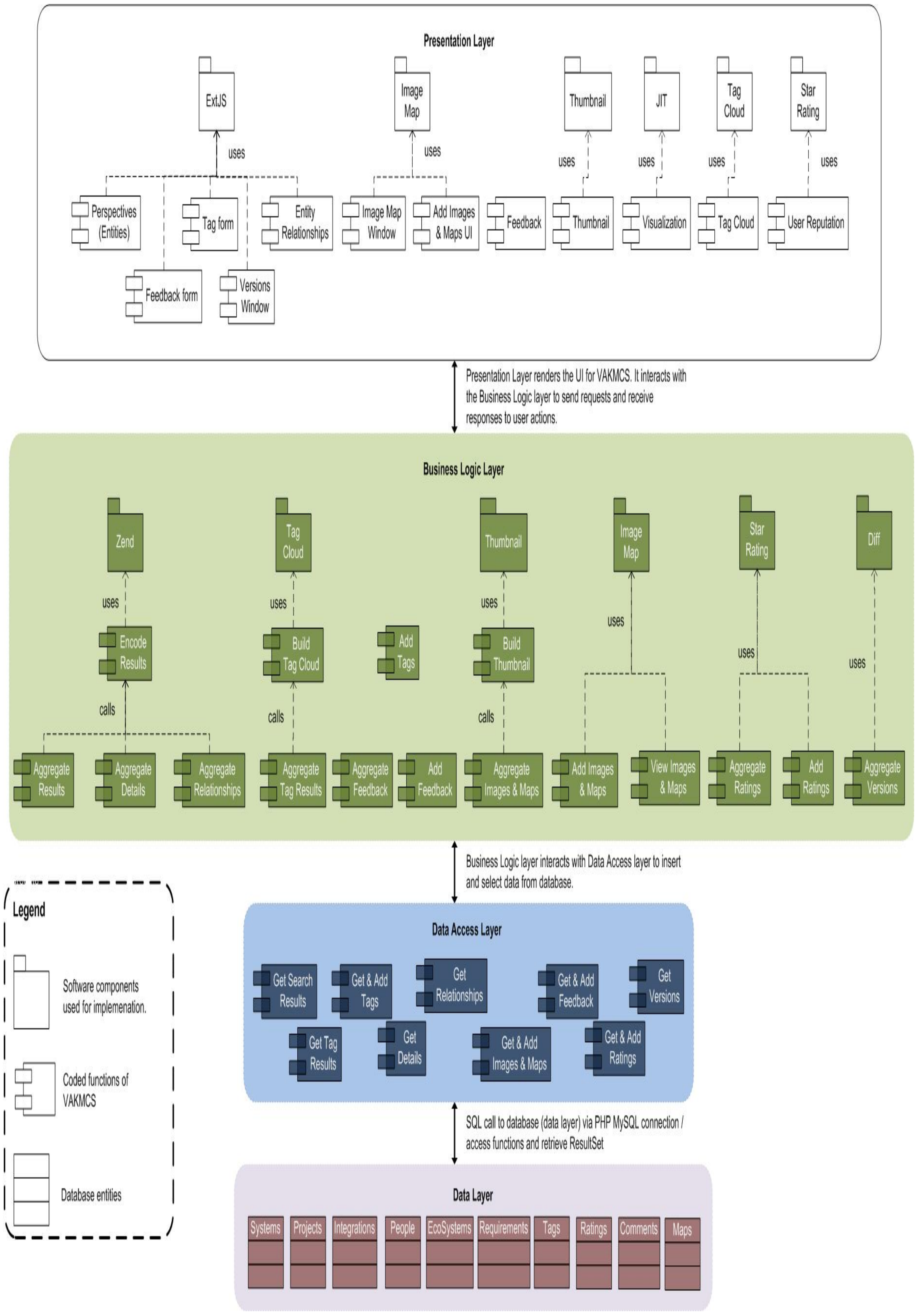


Figure 18. Software architecture of VAKMCS

4.1 DATABASE SCHEMA

Below is the table that defines database schema for the VAKMCS application. The application uses this schema to support all the features listed in section three of the document.

Table Name	**Details**
Ratings	This table contains the relationship between a user and the individual rating that he/ she received from other users of the tool. The cardinality of the relationship between user and his/ her rating is zero to many.
tComments	This table contains the relationship between a system, ecosystem, requirement or integration and its feedback. The cardinality of the relationship for each entity to its feedback is zero to many.
tEcoSystems	This table contains the master details associated with an Ecosystem. The table contains a hierarchy within itself of the versions and its parent's version.
tImageLocation	This table stores details regarding an image used for image mapping. It contains the server location of the image once it has been updated by a user. The cardinality of the relationship for each image within tImageLocation is one to one for the image map information in tImages tables.
tImages	This table contains the image map (coordinates within an image associated with a tag) generated for an image once a user tags.
tImagesTags	This table contains the individual tags associated with the images. The cardinality of the tags is many to one with images.
tIntegrations	This table contains the master details associated with an integration. The table contains a hierarchy within itself of the versions and its parent's version.
tIntegrations_EcoSystems	This table contains the integrations associated with ecosystems. The cardinality of this relationship is one ecosystem to many integrations.
tIntegrations_Projects	This table contains the integrations associated with projects. The cardinality of this relationship is one project to many integrations.
tPeople	This table contains basic information regarding a user. In a real world situation the user information would be integrated with an internal directory like LDAP.
tProjects	This table contains the master details associated with a project. The table contains a hierarchy within itself of the versions and its parent's version.

tProjects_People_Role	This table contains a relationship between a project and the roles that people played within it. The cardinality is one project to many people to one role.
tRequirements	This table contains the master details associated with a requirement. The table contains a hierarchy within itself of the versions and its parent's version.
tRequirements_Integrations	This table maintains the relationship between integration and its requirements. The cardinality between is one integration to many requirements.
tRequirements_Systems	This table maintains the relationship between systems and its requirements. The cardinality between is one system to many requirements.
tRoles	This table contains basic information regarding a role a user played with a project.
tSystems	This table contains the master details associated with a system. The table contains a hierarchy within itself of the versions and its parent's version.
tSystems_EcoSystems	This table maintains the relationship between system and the ecosystems it belongs to. The cardinality between is one system to many ecosystems.
tSystems_Integrations	This table maintains the relationship between systems and its integrations. The cardinality between is one system to many integrations.
tSystems_Projects	This table maintains the relationship between systems and the projects it belongs to. The cardinality between is one system to many projects.
tTags	This table maintains the relationship between any entity (system, project, ecosystem, integration and requirement) and the tag as added by a user. One entity has one-to-many relationship with tags.

Table 5. A table containing the details associated with the data layer of VAKMCS

Below are the entity relationship (ER) diagrams for the VAKMCS. The first diagram depicts the relationships of the entities (users, systems, requirements, projects, integrations and ecosystems) to each other and second diagram depicts how the images, tags and feedback are related to entities.

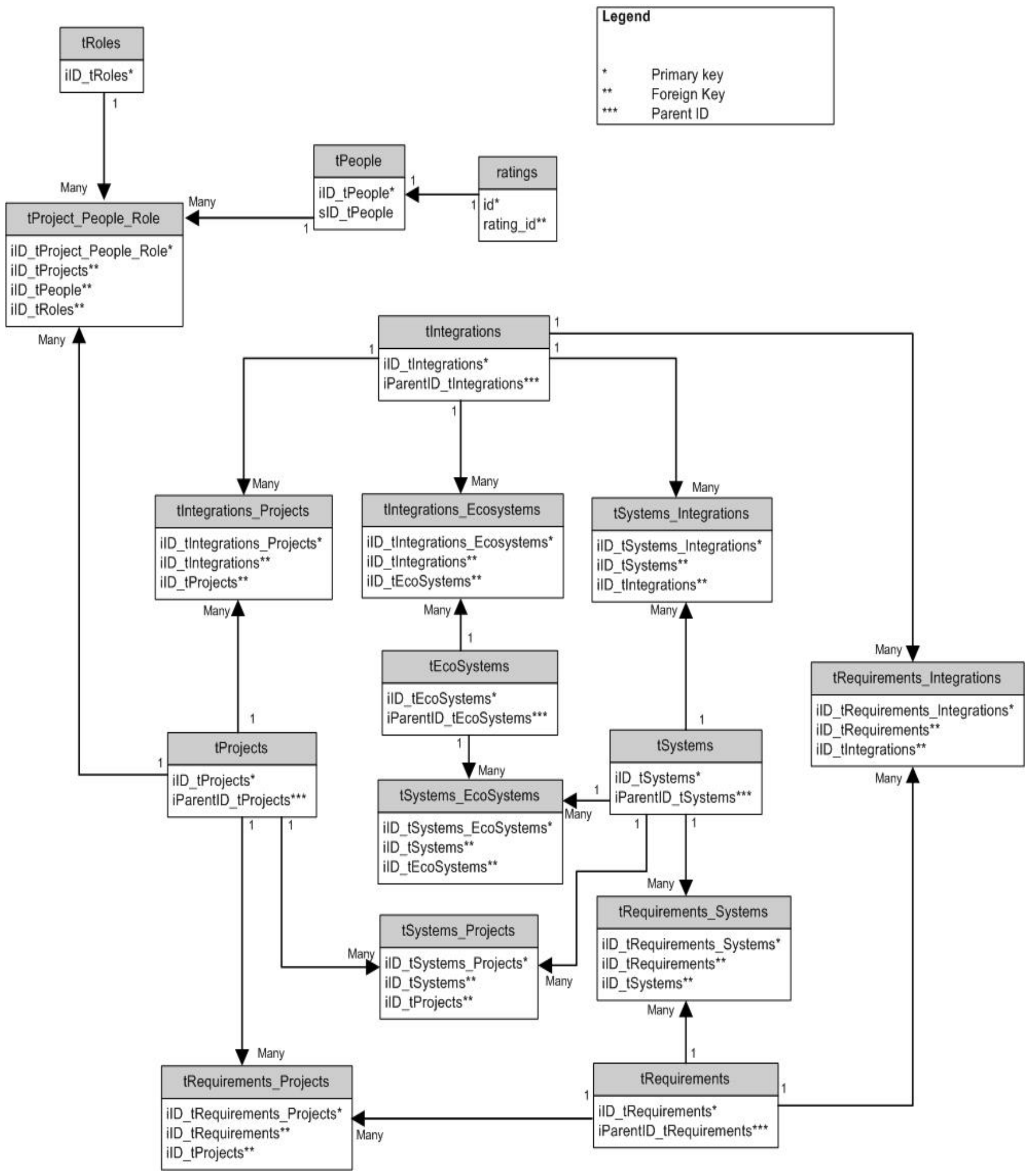


Figure 19. An ER diagram of VAKMCS entities and their relationships

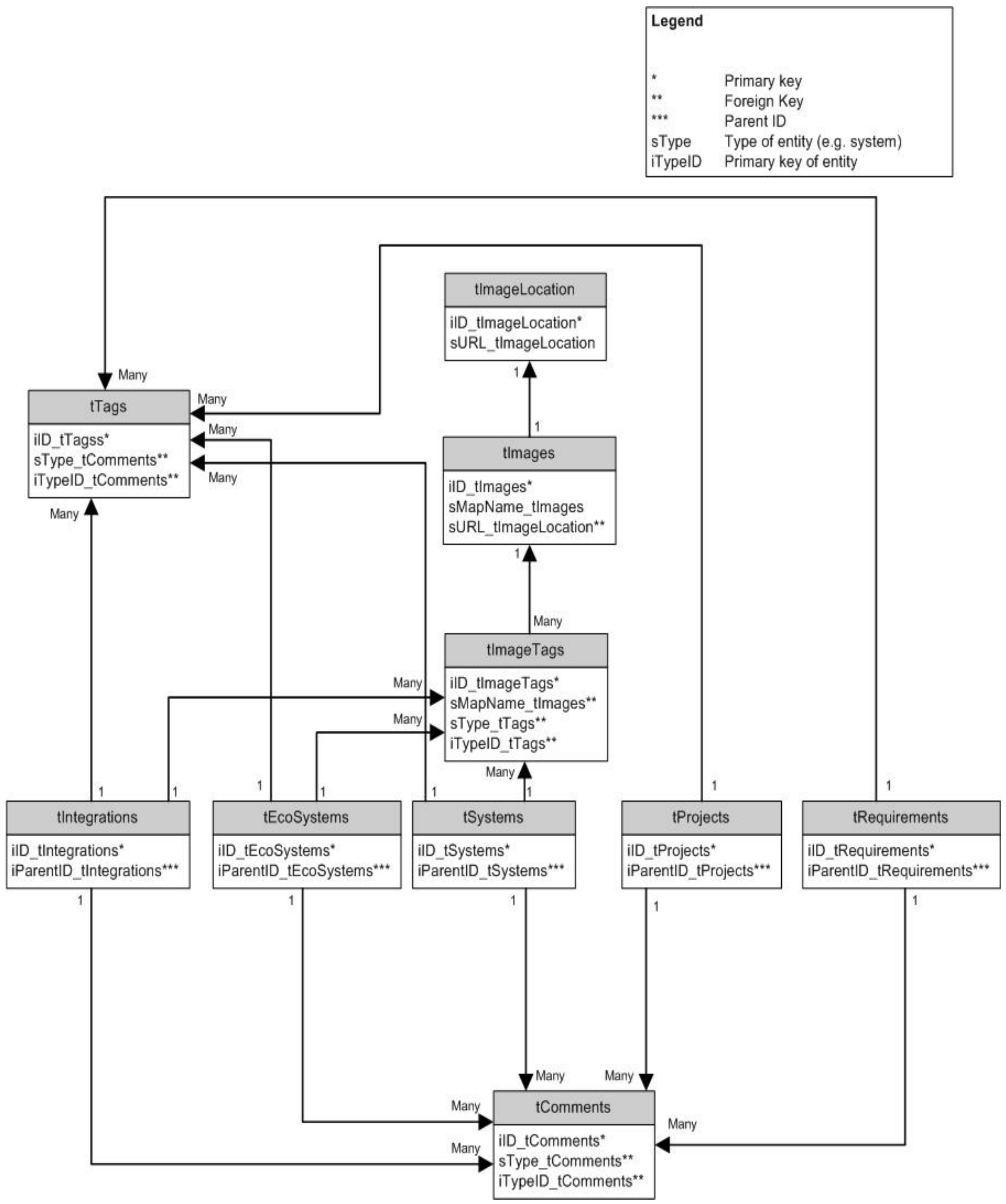


Figure 20. An ER diagram of VAKMCS entities and tags, images & feedback

4.2 DATA ACCESS LAYER

Each entity (e.g. systems) uses the data access layer with a similar set of access functions that enable the application (VAKMCS) to acquire related data from the perspective of a particular entity. The set of functions that each entity leverages for acquiring related data is described below in the table. Each function is called by the business layer in response to an action by the user of VAKMCS. The table below breaks down the data access by function, action and definition. The italics in the function name are representative of where entity name is substituted within the code. E.g. get*Entity*BySearchByLimit represents getSystemBySearchByLimit, getIntegrationBySearchByLimit etc.

Function	**Action**	**Definition**
get*Entity*bySearchByLimit	Search by text match	Returns the entities that match the search string. The result set is limited to 30 results at a time to allow for paginated result display.
get*Entity*Details	Get details on entity	Returns the details associated with the entity. Depending on the entity the details may differ slightly.
get*Entity*byTagbyLimit	Browse by tag match	Returns the entities that have been tagged with the text that matches the browsed tag. The result set is limited to 30 results at a time to allow for paginated result display.
get*RelatedEntity*By*Entity*By Limit	Get details on entity	Returns entities related to a specific entity. E.g. getSystemsByEcoSystemsByLimit, where the function returns all systems associated with the ecosystem.
getTagsBy*Entity*Tag	Browse by tag match	Returns the tags for all entities that match based on the text of the tag used to browse.
getTagsBy*Entity*Search	Search by text match	Returns the tags for all entities that match based on the text of the search criteria.
getMapNamesBy*Entity*	Get details on entity	Returns the map names that match a particular entity. This map is used to

		render the image mapping feature.
getImagesByEntity	Get details on entity	Returns images associated with an entity. The map names and images are related to enable the image mapping feature.
getPrevious*Entity*Version	Get details on entity	Returns the previous version of an entity if any.
getNext*Entity*Version	Get details on entity	Returns the next version of an entity if any.
getTagsBy*RelatedEntity*By*Entity*	Get details on entity	Returns the tags for all related entities.
get*Entity*Comments	Get details on entity	Returns the comments associated with entity.

Table 6. A list of common functions across the entities found in the data access layer

4.3 BUSINESS LOGIC & PRESENTATION LAYER

The business logic and the presentation layer of the application contain the relationships and business rules that are not accommodated by the data layer. The application uses these layers to encapsulate all relevant data based on entity and action to render for the user. The presentation layer leverages the ExtJS JavaScript library functions to make asynchronous calls to the server side PHP code (business logic) which assimilates the data gathered from the data access layer. The PHP code performs four main types of functions; gather information to display on index page, gather information to display as part of search results on each tab (for each entity), gather information to display as part of a browse results on each tab (for each entity) and gather information to display as part of a drill down for details regarding a specific entity.

Below is a diagram that depicts the functions of the business logic and presentation layer based on types of pages rendered by the application.

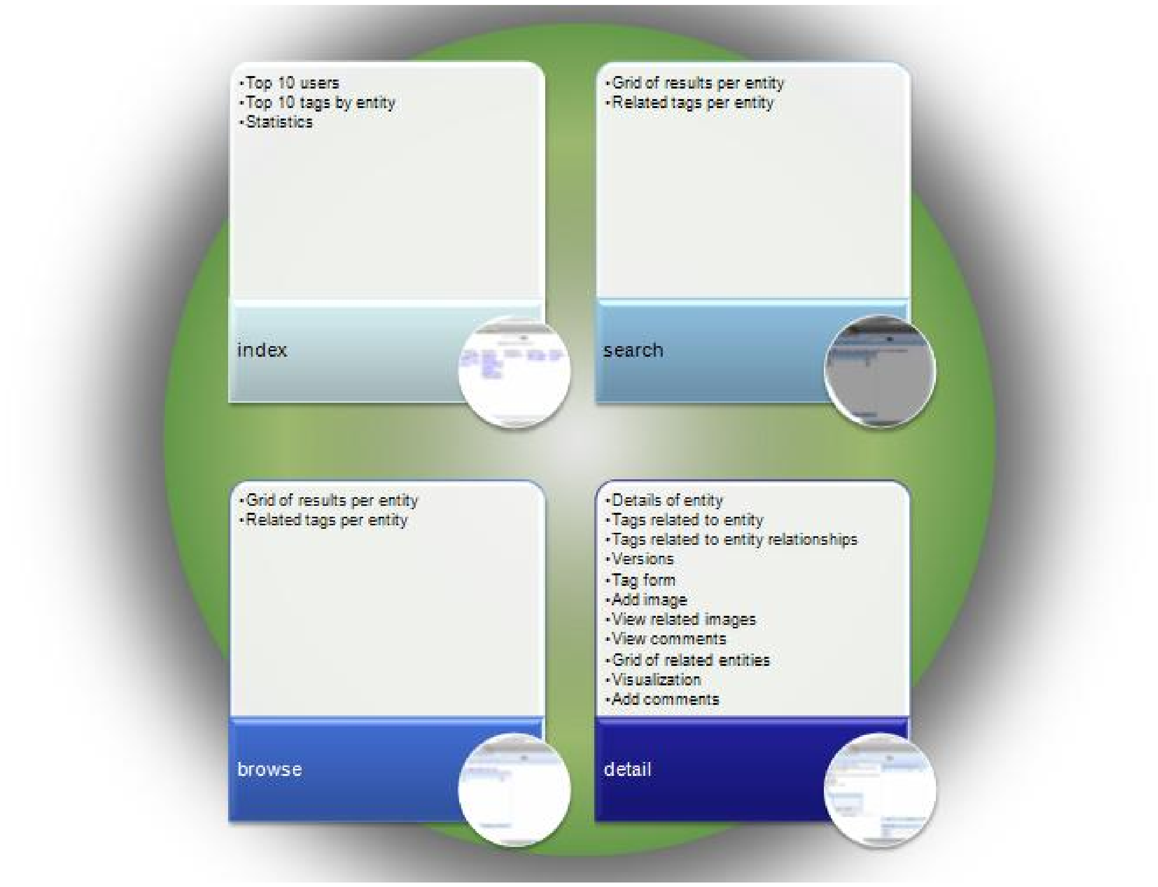


Figure 21. A block diagram of the business logic and presentation layer of VAKMCS

4.4 VAKMCS SOFTWARE DESIGN

VAKMCS leverages the ExtJS UI development kit to render the user interface, results and tagging and feedback capabilities. The basic layout of the application uses tabbed view of the entities (system, requirements etc.) and renders results within the appropriate tab. ExtJS framework supports asynchronous calls to server side code which VAKMCS implements for certain functionality and features. By rendering the results of a search/ browse or results of related entities within an ExtJS grid, VAKMCS leverages the framework to fire events based on the row selected. Based on the actions by the user, either all tabs are updated (e.g. search) or a certain tab is updated (e.g. browse and details). Both the feedback and tagging capabilities leverage the ExtJS forms to asynchronously store the details added by a user.

VAKMCS also integrates multiple other tools to enable features like user reputation, image mapping and visualization. In the following subsections of the document significant components and their design will be described through diagrams. The diagrams display key tasks performed as part of the flow, as well as, application layers in which the leveraged/ reused software components are used to implement VAKMCS features.

4.4.1 TAG CLOUDS

As stated in section three of this document, VAKMCS provides insight into the architecture information stored by providing the ability to browse by tags related to an entity. The design of this feature leveraged the software tool tag cloud. Tag cloud renders a cloud of tags based on the text and a quantity: the greater the quantity for a particular tag, the greater the size of the text of the tag within the cloud. The user has the option to click on the tag rendered to browse an entity. As an example, the diagram below depicts software flow for rendering top ten tags on the index page of VAKMCS.

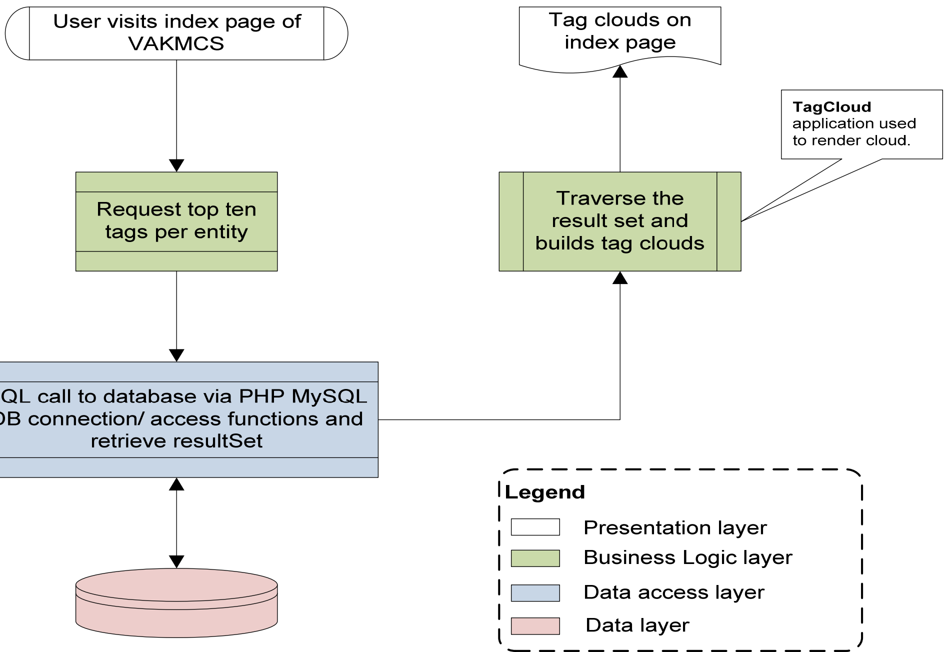
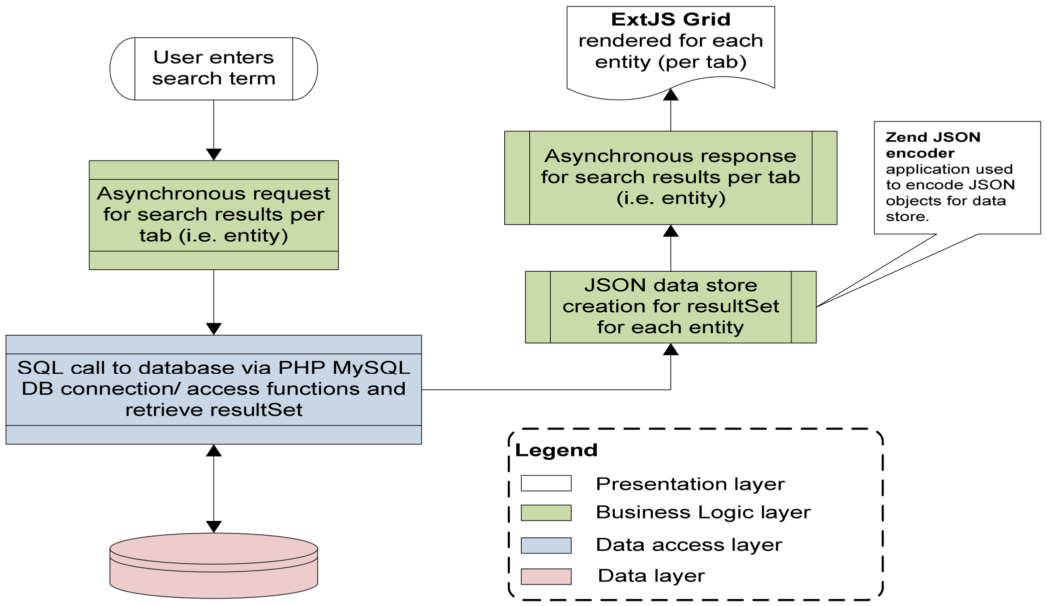

User visits index page of VAKMCS
Request top ten tags per entity
SQL call to database via PHP MySQL DB connection/ access functions and retrieve resultSet
Traverse the result set and builds tag clouds
TagCloud application used to render cloud.
Tag clouds on index page
Legend
Presentation layer
Business Logic layer
Data access layer
Data layer

Figure 22. Top ten tags flow

4.4.2 SEARCH RESULTS

The search feature in VAKMCS renders the results for all entities in one call to the server. This feature is enabled by using a grid within a tabbed layout from the ExtJS JavaScript development kit. The search terms are matched within attributes of the architecture information (using data access layer) and the results are encoded using Zend JSON encoder (software tool). The ExtJS grid asynchronously calls the server side business logic to create a data store (ExtJS uses the JSON encoded data) and renders the results upon callback. The diagram below displays the flow for rendering the search results on the search page in VAKMCS.

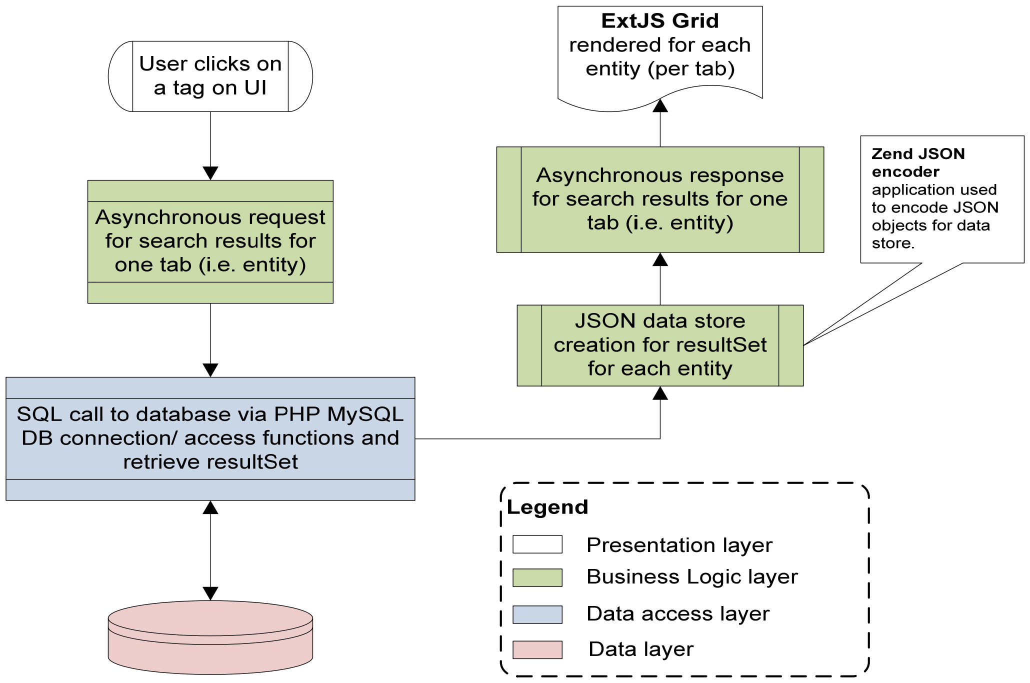


User enters search term
Asynchronous request for search results per tab (i.e. entity)
SQL call to database via PHP MySQL DB connection/ access functions and retrieve resultSet
JSON data store creation for resultSet for each entity
Asynchronous response for search results per tab (i.e. entity)
ExtJS Grid rendered for each entity (per tab)
Zend JSON encoder application used to encode JSON objects for data store.
Legend
Presentation layer
Business Logic layer
Data access layer
Data layer

Figure 23. Search flow

4.4.3 BROWSE RESULTS

The browse functionality provided by VAKMCS enables the user to find architecture information based on community driven context. This feature renders the results in a similar fashion to that of a search, but only updates one entity (one tab on the user interface) at a time with the result set. The design approach to realize this feature leveraged ExtJS grid and Zend JSON encoder as used in search. The diagram below displays the flow for rendering the browse results on the browse page in VAKMCS.



ExtJS Grid rendered for each entity (per tab)
User clicks on a tag on UI
Asynchronous response for search results for one tab (i.e. entity)
Zend JSON encoder application used to encode JSON objects for data store.
Asynchronous request for search results for one tab (i.e. entity)
JSON data store creation for resultSet for each entity
SQL call to database via PHP MySQL DB connection/ access functions and retrieve resultSet
Legend
Presentation layer
Business Logic layer
Data access layer
Data layer

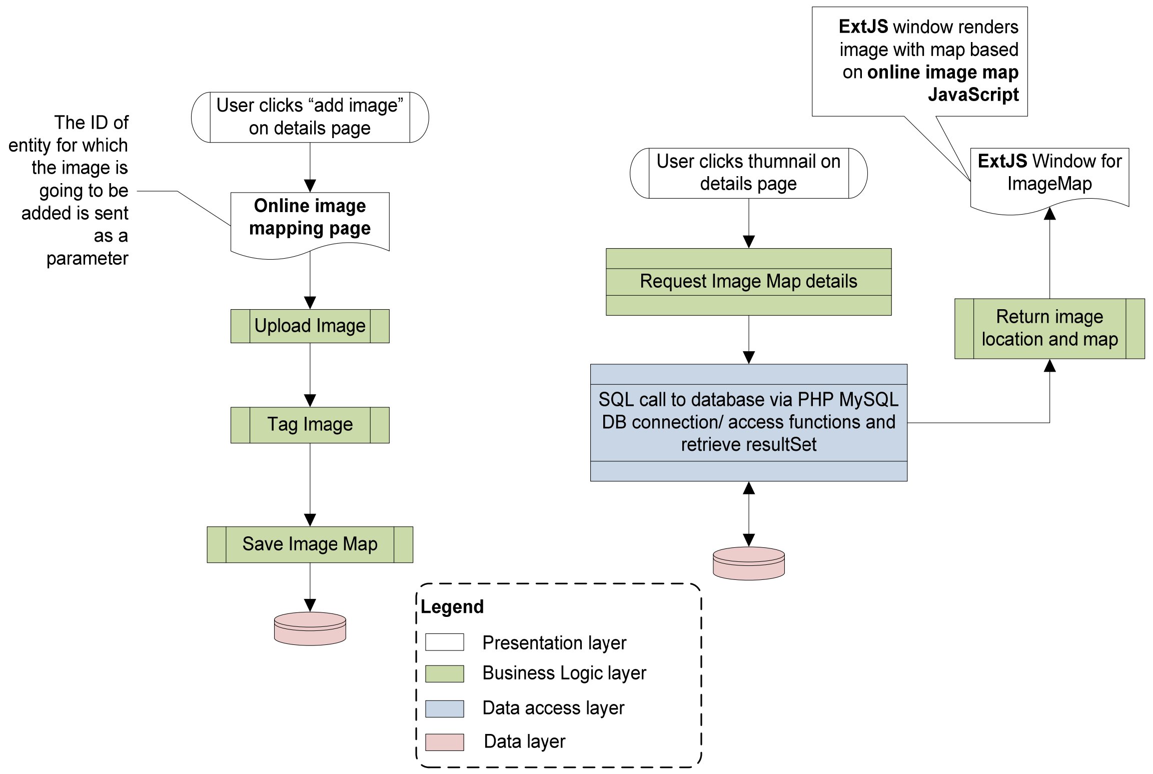
Figure 24. Browse flow

4.4.4 IMAGE MAPPING

The image mapping feature of VAKMCS enables two functions: the first function is the ability to add an image map related to a particular entity (e.g. system or integration), while the second is the ability to view and interact with the image map. To enable the add feature required the integration of an online image map editor (software tool). To enable the view feature required leveraging ExtJS Window (development kit) functionality and thumbnail capabilities provide

In respect to the add image functionality, the tool offers the ability to upload an image and associated maps and context within the image. An image is associated with a map of coordinates and tags. Once the user completes the image mapping process and is ready to save the resulting map, the integration allows the ability to store the appropriate relationships, by relating to a particular entity by its ID (based on where the user chose to add an image) and others by tags within VAKMCS.

To view the map, the user navigates into the details page of a system, integration or ecosystem and finds thumbnails (software tool) of the images. A relationship between the image and the entities is based on either the ID of an entity (if a user actually added a picture specifically for that entity) or by matching the tags associated with the image and the entity. Upon clicking a thumbnail, an ExtJS Window pops-up to render the image and the associated tags for the user. The diagram below displays the flow for adding and viewing image maps in VAKMCS.



The ID of entity for which the image is going to be added is sent as a parameter
User clicks "add image" on details page
Online image mapping page
Upload Image
Tag Image
Save Image Map
ExtJS window renders image with map based on online image map JavaScript
User clicks thumnail on details page
ExtJS Window for ImageMap
Request Image Map details
Return image location and map
SQL call to database via PHP MySQL DB connection/ access functions and retrieve resultSet
Legend
Presentation layer
Business Logic layer
Data access layer
Data layer

Figure 25. Image add (left) and view flow (right)

4.4.5 USER REPUTATION

User reputation as described in section three of this document is driven by star ratings given to users by contributors and reviewers of content within VAKMCS. This feature integrates the StarRating software tool and associates users with a rating. The ratings appear in associated with comments within VAKMCS on the details page. As the comments are rendered, each author's rating (calculated as an average) is rendered along with the comment. The tool allows adding a rating to a user by an asynchronous JavaScript call to the server side, upon completion, the rating on the details page reflects the latest average. The diagrams below display the flow for add and viewing user reputation on the details page in VAKMCS.

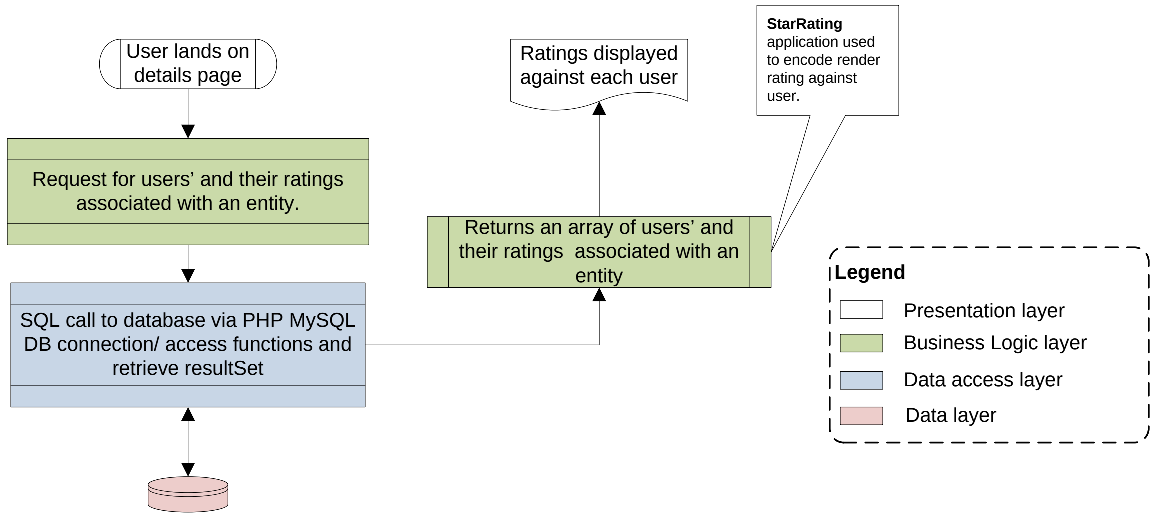

User lands on details page
Request for users' and their ratings associated with an entity.
SQL call to database via PHP MySQL DB connection/ access functions and retrieve resultSet
Returns an array of users' and their ratings associated with an entity
Ratings displayed against each user
StarRating application used to encode render rating against user.
Legend
Presentation layer
Business Logic layer
Data access layer
Data layer

Figure 26. View user reputation flow

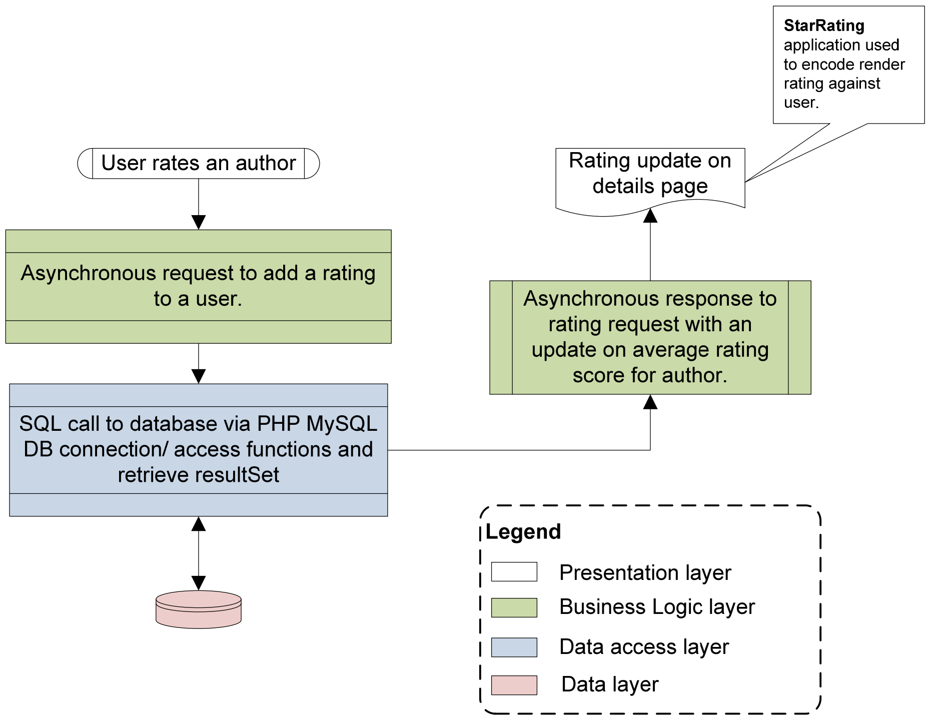

StarRating application used to encode render rating against user.
User rates an author
Rating update on details page
Asynchronous request to add a rating to a user.
Asynchronous response to rating request with an update on average rating score for author.
SQL call to database via PHP MySQL DB connection/ access functions and retrieve resultSet
Legend
Presentation layer
Business Logic layer
Data access layer
Data layer

Figure 27. Add user rating flow

4.4.6 VISUALIZATION

Visualization function in VAKMCS builds a graph based on an entity (project or integration) and its relationships. It offers the user an ability to review relationships in a moving visual format rather than a static table format. The visualization implementation leverages JIT (JavaScript library) to render the graph. The implementation involved organizing related data based on categories of relationships (e.g. for project: team members, systems, integrations etc.) as accessed through the data layer and then encoding it using Zend JSON encoder to finally rendering it using the JIT functions. The diagram below portrays the visualization flow.

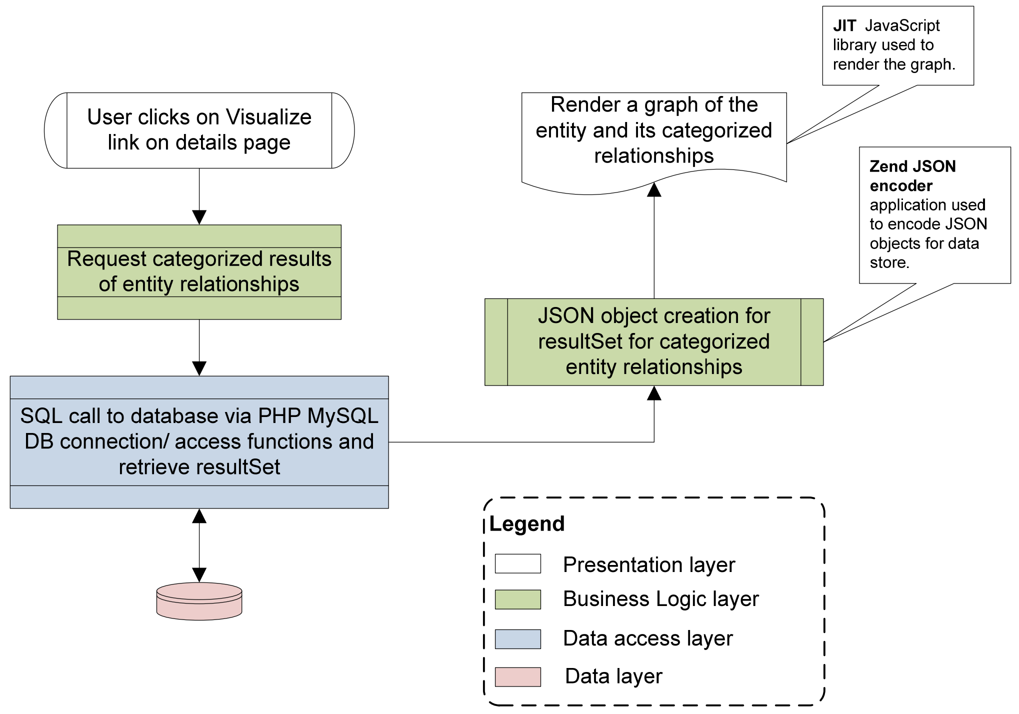

User clicks on Visualize link on details page
Request categorized results of entity relationships
SQL call to database via PHP MySQL DB connection/ access functions and retrieve resultSet
JSON object creation for resultSet for categorized entity relationships
Render a graph of the entity and its categorized relationships
JIT JavaScript library used to render the graph.
Zend JSON encoder application used to encode JSON objects for data store.
Legend
Presentation layer
Business Logic layer
Data access layer
Data layer

Figure 28. Visualization flow

4.4.7 VERSIONS

The versions feature of VAKMCS compares the current entity being viewed in the details page with its previous and next versions if available/ authored. The implementation collects all information associated with an entity and its versions into three separate arrays and then leverages an array comparison script to provide the differences. The differences are rendered on the UI through an ExtJS window upon user clicking the “Versions” hyperlink on the details page. The diagram below portrays the version feature flow.

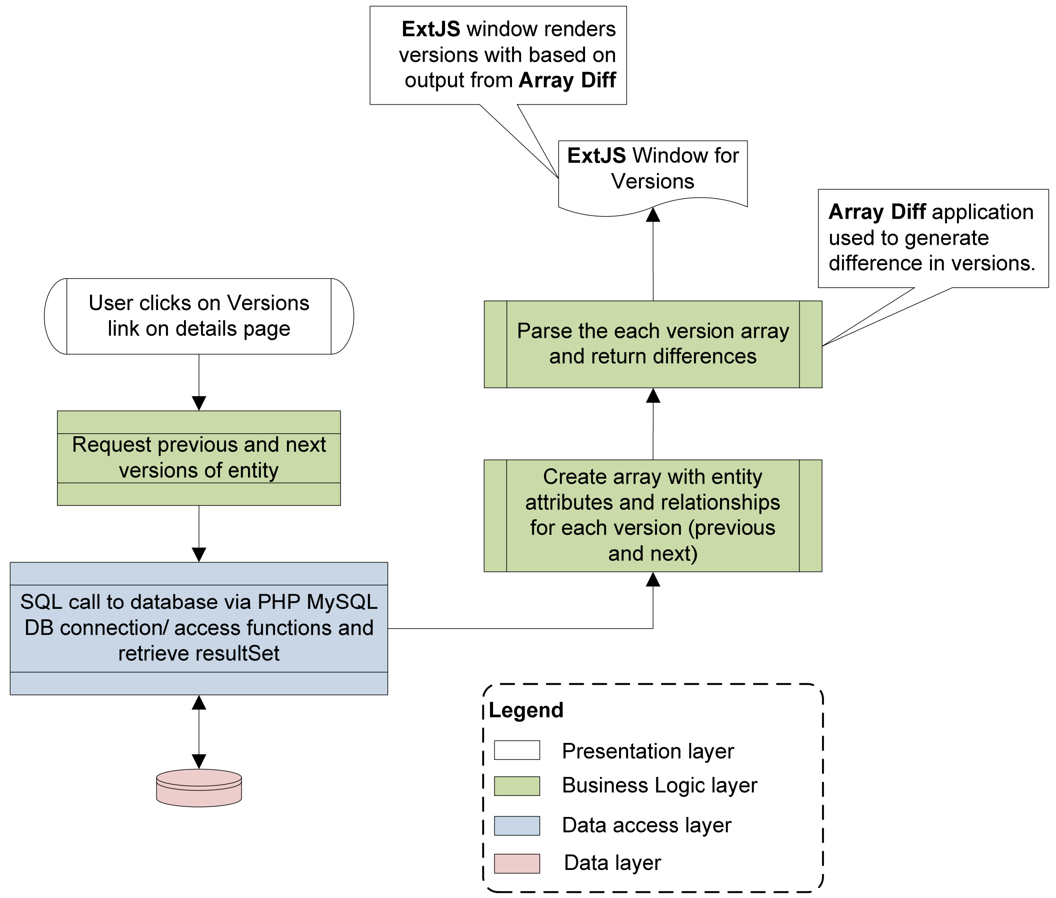

ExtJS window renders versions with based on output from **Array Diff**
ExtJS Window for Versions
Array Diff application used to generate difference in versions.
User clicks on Versions link on details page
Parse the each version array and return differences
Request previous and next versions of entity
Create array with entity attributes and relationships for each version (previous and next)
SQL call to database via PHP MySQL DB connection/ access functions and retrieve resultSet
Legend
Presentation layer
Business Logic layer
Data access layer
Data layer

Figure 29. Versions flow

5.0 SOFTWARE TOOLS, DEVELOPMENT KITS USED

Below is a list of components and their sources as utilized during the implementation of the VAKMCS project. Also noted are the software architecture layers (as described in the previous section of the document) where these components are utilized.

- Application for image uploading/ image mapping:

  http://www.maschek.hu/imagemap/imgmap

  This component is utilized in both the presentation and the business logic layers. The user interface for adding an image map and viewing an image map are supported through HTML and JavaScript while a PHP/ MySQL backend support the storage and retrieval of the image and map itself.

- Application for ratings: http://boedesign.com/posts/23.html

  This component is utilized in both the presentation and the business logic layers. The user interface for viewing and adding a rating are supported through JavaScript while a PHP/ MySQL backend support the storage and retrieval of ratings.

- Application for building tag-clouds based on tagging parameters:

  http://www.lotsofcode.com/php/tutorials/tag-cloud

  This component is utilized in both the presentation and the business logic layers. The user interface for viewing the tag clouds provides a style sheet and logic for rendering the HTML cloud with varying sizes based on count, while the backend PHP/ MySQL supports the retrieval of the tags themselves.

- JavaScript UI SDK: http://www.extjs.com

  This user interface framework is utilized to render the layout and many of the input and output user interface components of the presentation layer of VAKMCS. The framework supports rendering output based on asynchronous calls to the PHP/ MySQL backend for retrieval of data in JSON format. The framework supports storing user inputs through forms by sending asynchronous calls to the PHP/ MySQL backend. The framework provides a vast library of out of box user interface components including grids, forms, tabs and layouts as implemented within VAKMCS for several features.

- Visualization JavaScript graphing library: http://thejit.org

  This component is utilized in the presentation layer. The user interface for viewing and interacting with the graph is supported by JavaScript fed through the backend data by PHP/ MySQL in JSON format.

- Application for creating image thumbnails: authored by Ian Selby (ian@gen-x-design.com)

  This component is utilized in both the presentation and the business logic layers. The user interface for viewing the thumbnail provides a style sheet and logic for rendering the HTML and images while the backend PHP/ MySQL supports the resizing of the images themselves.

- Application for encoding and decoding JSON: http://www.zend.com

  This component is utilized in the business logic layer. The library enables data to be encoded into JSON based on nested array definition. VAKMCS utilizes this feature to render data in the requested format of the other software components used in the implementation including visualization library and ExtJS framework.

- Application for version difference annotation: authored by Daniel Unterberger (d.u.diff@holomind.de)

  This component is utilized in the business logic layer. The library enables the evaluation of the difference of two arrays by annotating the differences. This feature is used as part of the versioning function of VAKMCS.

6.0 VAKMCS VS. PAKME

This section of the document will disclose the implementation and design differences between PAKME and VAKMCS. By providing a concise breakdown of the differences, this paper will provide insight into the innovative aspects of VAKMCS. Though multiple architecture knowledge management solutions were reviewed during the discovery phase of the project, the paper focuses on PAKME for comparison as it was one of the most complete implementations.

In the following sections significant differences in approach and function will be discussed to provide a comparison.

6.1 SOFTWARE COMPONENTS

The PAKME solution is built on top of an open source groupware platform, Hipergate [1] that is extended to provide features for architecture knowledge storage and retrieval. VAKMCS is built ground up leveraging and integrating existing point solutions (e.g. star rating)/ frameworks to provide multiple perspectives into architecture data. VAKMCS was built from scratch to allow freedom for implementation limitations found in leveraging open source platforms. Another reason for choosing a bottom up route was to change the perspective of the application functionality from technology centric information to business centric information.

6.2 KNOWLEDGE DEFINITION

One of the core differences between PAKME and VAKMCS is the way that architecture knowledge is defined within the tool. PAKME defines knowledge by scenarios, requirements, quality factors, analysis model, patterns, architecturally significant requirement, architecture decision, alternative decision and findings.

VAKMCS defines knowledge by systems, requirements, integrations, ecosystems and projects. The differentiation in approach is to accommodate different utilizations of the knowledge. PAKME focuses more on providing access to technology centric information, by enabling designers to use accumulated “wisdom” from different projects when devising or analyzing architectural decisions [1], while VAKMCS focuses on business centric information by providing search friendly access to existing solutions to decide whether spending is needed for a new IT project.

6.3 FEATURE COMPARISON

Below are a set of features that differentiate PAKME and VAKMCS. Features for search, versions are common to both platforms, however their implementations differ. Features for browse, feedback, user reputation, image tagging and visualization are not found in PAKME, but implemented in VAKMCS. Details regarding the VAKMCS features lacking in PAKME have already been provided in section three, hence, a brief outline of the implementation difference on shared features is provided in the following subsections.

Feature	**PAKME**	**VAKMCS**
Motivation: enable functions that motivate users to participate and author content within architecture system.		✓
Search: enable text based search of stored architecture knowledge.	✓	✓
Textual Tagging: enable community driven context of stored architecture knowledge.		✓
Image Tagging: enable community driven context of images.		✓
Visualization: enable visual and graphical representation of architecture knowledge and its relationships.		✓
Feedback: enable user feedback for architecture knowledge.		✓
User Reputation: enable rating system to understand contributor of knowledge.		✓
Versions: enable comparison of past and future iterations	✓	✓

of architecture knowledge.		
Browse: enable category based search of architecture knowledge.	✓	✓

Table 7. Basic comparison of VAKMCS and PAKME

6.3.1 SEARCH / BROWSE

PAKME and VAKMCS both perform search by text matches, however, PAKME provides the ability to search by selecting the data element (e.g. title within a general scenario) while VAKMCS searches the title and description fields of the entities. PAKME also has a slightly more intelligent search functionality that allows the ability to search by using bitwise logic. VAKMCS did not implement this functionality as it could be tackled with search engine integration.

PAKME does not support categorized browse functionality, this limits the ability for users to have both community driven context, as well as, the ability to have some “hints” in how to search for a particular knowledge. VAKMCS leverages tagging to provide users with a starting point for their search and should help the user find relevant information more easily than PAKME.

6.3.2 VERSIONS

Both PAKME and VAKMCS provide the ability to track versions, however, PAKME does not provide a quick view into the differences between two versions. This feature allows users to have an insight into the past and future roadmap of an entity and hence give some idea regarding possible alignment between business functions and lead to leveraging existing solution rather than investing on a new implementation.

7.0 VAKMCS VS. ADDSS

This section of the document will disclose the implementation and design differences between ADDSS and VAKMCS. By providing a concise breakdown of the differences, this paper will provide insight into the innovative aspects of VAKMCS. Over all, ADDSS falls short in its implementation of the proposed features. The tool is not intuitive in usage and still has many features not implemented. Based on the available literature and some hands on usage the following sections will show the differences in approach and function between the two systems.

7.1 SOFTWARE COMPONENTS

Both ADDSS and VAKMCS are built ground up, and use PHP and MySQL. While VAKMCS leverages ExtJS framework for its layout, ADDSS does not use any UI framework. Both ADDSS and VAKMCS have used thumbnail libraries for images, however the purposes of the images are different.

7.2 KNOWLEDGE DEFINITION

As with PAKME, one of the core differences between ADDSS and VAKMCS is type of architecture knowledge stored within the tool. ADDSS focuses on storing information regarding design decisions and iterations in the decision making process, while VAKMCS caters to knowledge by systems, requirements, integrations, ecosystems and projects. The differentiation in approach is to accommodate different utilizations of the knowledge. Like PAKME, ADDSS focuses more on providing access to technology centric information, by enabling designers to use accumulated “wisdom” from different projects when devising or analyzing architectural decisions [1], while VAKMCS focuses on business centric information by providing search friendly access to existing solutions to decide whether spending is needed for a new IT project.

7.3 FEATURE COMPARISON

Below are a set of features that differentiate ADDSS and VAKMCS. Features for versions and browse are common to both platforms, however their implementations differ. Features for search, feedback, textual tagging, user reputation, image tagging and visualization are not found in ADDSS, but implemented in VAKMCS. Details regarding the VACKMS features lacking in ADDSS have already been provided in section three, hence, a brief outline of the implementation difference on shared features is provided in the following subsections.

Feature	**ADDSS**	**VAKMCS**
Motivation: enable functions that motivate users to participate and author content within architecture system.		✓
Search: enable text based search of stored architecture knowledge.		✓
Textual Tagging: enable community driven context of stored architecture knowledge.		✓
Image Tagging: enable community driven context of images.		✓
Visualization: enable visual and graphical representation of architecture knowledge and its relationships.		✓
Feedback: enable user feedback for architecture knowledge.		✓
User Reputation: enable rating system to understand contributor of knowledge.		✓
Versions: enable comparison of past and future iterations of architecture knowledge.	✓	✓
Browse: enable category based search of architecture knowledge.	✓	✓

Table 8. Basic comparison of VAKMCS and ADDSS

7.3.1 BROWSE

Like PAKME, ADDSS does not support categorized browse functionality, and thus limits the ability for users to have both community driven context, as well as, the ability to have some “hints” in how to search for a particular knowledge.

7.3.2 VERSIONS

Version management feature of ADDSS has better capabilities than VAKMCS. The application provides multiple characteristics by which to differentiate a version. The application also provides a means to view a chronological breakdown of the version from inception. VAKMCS only uses simple descriptions and relationships to measure version changes at time, as described in future project sections, there are opportunities for enhancements in the future.

8.0 POSSIBLE FUTURE WORKS

8.1 CHAT & EMAIL

Allow the ability for a user to connect directly through the tool to a project team member. By incorporating the communication between the team member of the project and the user within the application we can capture the details of the communication and make them another source of knowledge.

8.2 SEARCH ENGINE

By adding a search engine layer between the data access and the database layer one can improve the performance of the data access. Indexes can be built on new relationships that are currently not functions in VAKMCS, e.g. mapping multiple search terms (criteria) to multiple related entities at once. Other benefits of implementing a search engine integration would include, but not be limited to, content generated categories (e.g. search by common words within title and/ or description for certain entities), and bitwise search.

8.3 VERSIONS

Versioning functions can be extended to have intelligence to add multiple features and knowledge that can be leveraged outside of what VAKMCS offers. One of the key areas of versioning that might be helpful for users of the system would be an intelligent crawler that can use knowledge semantics to decide to relate entities that are not explicitly connected, i.e. enhance versioning software to do intelligent difference calculations to find newer versions of a particular entity. Another feature related to versioning that might be helpful would be allowing the ability to compare images that are related to see visually the changes in a particular entity over time. This feature could help to architecturally understand the roadmap of the ecosystems.

8.4 VISUALIZATION

Currently visualization feature in VAKMCS provides the ability to access only immediately related information regarding projects. An extension of this feature would be to allow the visualization to change perspective by allowing multiple actions e.g. a user should be allowed to change the diagram by entering search terms and perspective (e.g. system, integration etc.).

8.5 REPORTING

Adding a function to generate reports (e.g. PDF format) with consolidated details regarding an entity would be helpful in understanding the complete solution. Enhancing VAKMCS with such a feature would also help users communicate outside of the tool by printing such a report, or leveraging it for a presentation.

8.6 USER & IMAGE PERSPECTIVE

Currently VAKMCS does not afford the feature to search browse and perform other functions on image and user functions like the entities. By enabling the image perspective functionality, users would have a broader range of perspectives to choose from and may decide to begin their search from a visual context (mapped images) rather than a textual context. The user perspective feature would cater allowing the community to track user participation within VAKMCS which could lead building features that help to understand a user’s subject matter expertise and allow further opportunities to collaborate.

9.0 POSSIBLE REAL WORLD IMPLEMENTATION

As the purpose of this project is to meet the Masters' requirements for graduation, the VAKMCS solution is a standalone approach to accommodate architecture knowledge management. In the real world data may not be stored in a single source of truth but rather spread across the enterprise. To support VAKMCS features in an enterprise would require more than just systems integration but also process, governance and administration. Below is diagram that depicts a possible approach for Cisco's integration of VAKMCS.

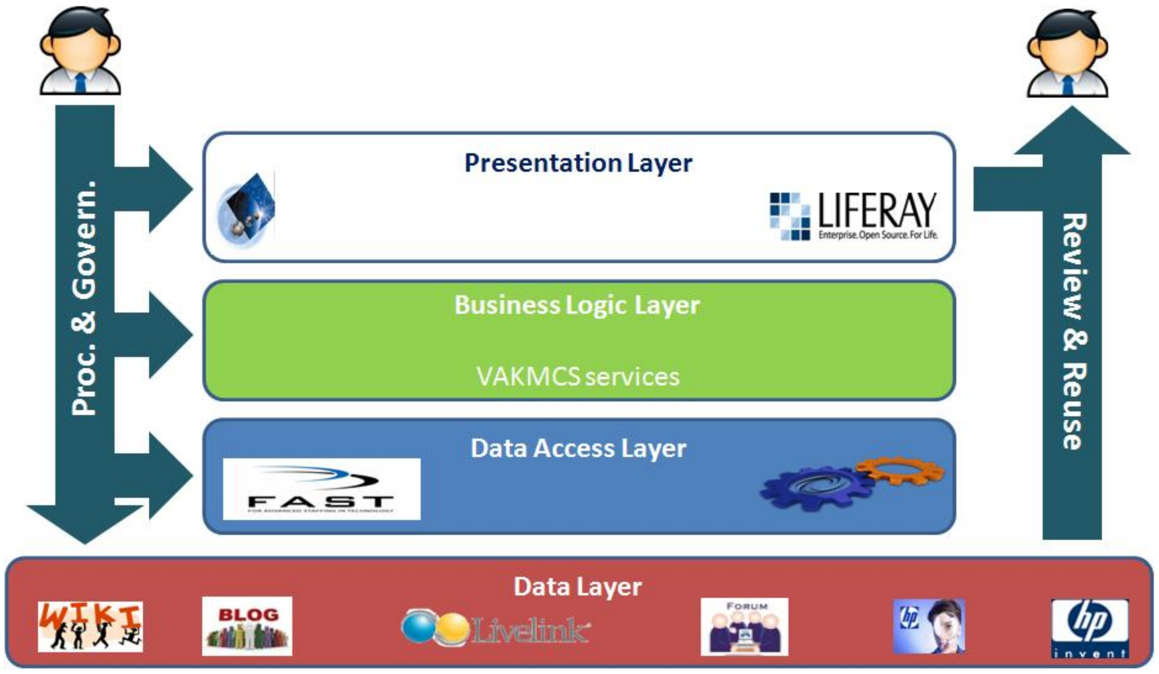


Figure 30. A possible VAKMCS integration at Cisco

As depicted in the diagram, Cisco already stores architecture information in various forms and tools. The data layer above shows some of these tools (left to right) as wiki, blog, document repository, forums, quality center for test cases and portfolio management tool. The storage and retrieval of the architecture knowledge from within one tool maybe incomplete or outdated, however, aggregating this knowledge in a meaningful would be the benefit of integrating VAKMCS. It is at this level that process and governance would be required to mandate certain data be recorded for existing and new projects so that the information is utilized in a meaningful manner. More details on the process, governance and administration are provided in the subsections.

The data access layer would consist of web services and search engine based integration to enable architecture information consumption and utilization by VAKMCS. There are multiple ways that Cisco can enable the access to content stored in the independent sources; however, standard practices include RSS and web service with XML/ JSON over HTTP.

The business logic layer would consist of the features provided by VAKMCS and support building meaning relationships and an aggregated knowledge of related entities. Unlike the implementation for the purposes of the Masters' requirements for graduation, in the real world, we could enhance VAKMCS to leverage only the business logic layer and interact directly with existing data and data access layers. We could also enhance VAKMCS to render the features as decoupled user interface elements (e.g. portlets, widgets) to be consumed by an external presentation layer.

The presentation layer could continue to leverage the user interface as implemented for this project; however, most enterprise's today use portals and mash-ups for presenting aggregated data across multiple sources. Based on experience, it is probable that Cisco would lean in the same direction when integrating VAKMCS.

The following subsections identify some others aspects of success for integrating VAKMCS into an enterprise like Cisco.

9.1 PROCESS

Building meaningful relationships utilizing VAKMCS would require an enterprise to set up a process for indentifying and enforcing standard criteria for architecture knowledge. The enterprise would need to regularly update these standards to support changes. The kinds of standards that an enterprise like Cisco may chose to categorize and relate their architecture data would include, but not be limited to, common terminology to identify system details, integration details, requirements, application service providers vs. internal applications etc. Enterprise evolution would constitute continued evolution and growth of VAKMCS and would require building a process for managing changes and release cycles for enhancements.

9.2 GOVERNANCE

A key factor to the success of integrating VAKMCS would be the governance and “top down” support from management. Initiating a significant change such as integrating VAKMCS would require education and adoption mandates from senior IT management to the individual contributors. Some of the appropriate actions that would be required by management include:

- A clear message on updating existing and creating new project documentation with standardized criteria would be required of all IT personnel.
- A mandate should be set by management to first review possible internal solutions leveraging VAKMCS before proposing a solution to meet the business needs.
- A program manager should be assigned to plan phases and updates for the VAKMCS features.

- A council of architects and knowledge management experts would be appropriate for updating standards as the enterprise evolves.
- A reward system should be implemented for recognizing positive contribution as determined through ratings.

10.0 REFERENCES

1. Babar M.A. & Gorton I. (May 2007). Architecture Knowledge Management: Challenges, Approaches, and Tools. *Companion to the proceedings of the 29th International Conference on Software Engineering ICSE COMPANION '07*.IEEE Computer Society

2. Barbar M.A. & Gorton I. (2007). A Tool for Managing Software Architecture Knowledge. *Second Workshop on Sharing and Reusing architectural Knowledge Architecture, Rationale, and Design Intent.* IEEE Computer Society.

3. Capilla R., Dueñas J. C., Nava F., & Pérez S. (2006). A web-based tool for managing architectural design decisions. *ACM SIGSOFT Software Engineering Note. Article No. 4.*

4. Becker M. & Falessi D. (2006). Documenting design decisions: A framework and its evaluation in the ambient intelligence domain. *IESE.* IESE-Report VII, 34 pp. : Ill., Lit. 050.06/E

5. Becker M., Cantone G. & Falessi D. (2006). Design decision rationale: experiences and steps ahead towards systematic use. *SESSION: SHAring and Reusing architectural Knowledge (SHARK '2006) paper abstracts.* ACM SIGSOFT Software Engineering Notes Volume 31, Issue 5.

6. Sunassee N.N. & Sewry D.A. (September 2003). An investigation of knowledge management implementation strategies. *Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology SAICSIT '03.* South African Institute for Computer Scientists and Information Technologists

7. Sunassee N.N. & Sewry D.A. (September 2002). Research papers: data/knowledge management: A theoretical framework for knowledge management implementation. *Proceedings of the 2002 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology SAICSIT '02.* South African Institute for Computer Scientists and Information Technologists

8. Stewart K.A., Baskerville R., Storey V.C., Senn J.A., Raven A. & Long C. (September 2000). Research contributions: Confronting the assumptions underlying the management of knowledge: an agenda for understanding and investigating knowledge management. *ACM SIGMIS Database, Volume 31 Issue 4.* ACM Press

9. Ezingeard J., Leigh S. & Chandler-Wilde R. (December 2000). Knowledge management at Ernst & Young UK: getting value through knowledge flows. *Proceedings of the twenty first international conference on Information systems ICIS '00.* Association for Information Systems

10. Hahn J. & Subramani M.R. (December 2000). A framework of knowledge management systems: issues and challenges for theory and practice. *Proceedings of the twenty first international conference on Information systems ICIS '00*. Association for Information Systems

11. Herrmann T., Hoffmann M., Jahnke I., Kienle A., Kunau G., Loser K. & Menold N. (November 2003). Knowledge management II: Concepts for usable patterns of groupware applications. *Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work GROUP '03.* ACM Press

12. Dingsøyr t., Djarraya H.K. & Røyrvik E. (December 2005). Practical knowledge management tool use in a software consulting company. *Communications of the ACM, Volume 48 Issue 12*. ACM Press

13. Agostini A., Albolino S., Michelis D.S., De Paoli F. & Dond Ri. Knowledge Management I: Stimulating knowledge discovery and sharing. (November 2003). *Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work GROUP '03*. ACM Press