THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

Architecture Knowledge Management: Challenges, Approaches, Tools

M. Ali Babar[1] & Ian Gorton[2]
[1]Senior Researcher, Lero, Ireland
[2]Chief Architect, Data Intensive Computing, PNNL, USA

Introductions

THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

- Tell us
  - Who you are
  - Educational / Professional Background
  - Research interests
  - Past experiences with architecture knowledge management
  - Expectations of tutorial

Tutorial Goals

- Comprehend the role and challenges involved in managing software architecture knowledge
- Understand domain modeling techniques for characterizing architecture knowledge
- Gain knowledge of the process for capturing and managing architecture knowledge
- Learn how to utilize architecture knowledge to improve the architecture design and evaluation process
- Appreciate the processes and tools required to build and manage a knowledge repository

Overview

- Section 1 – Architecture knowledge management
- Section 2 – A Tool and Approaches to Managing architecture knowledge
- Section 3 - Domain modeling for architecture knowledge
- Section 4 - Utilizing architecture knowledge during design and evaluation

Architecture Knowledge Management

- What is architecture knowledge?
- What is architecture knowledge management?
- Why is architecture knowledge management important?
  - Challenges
  - Strategies
  - Expected benefits
- State-of-the art
  - Tooling support
  - R&D
- Case study description

Architecture Knowledge and its Management

- Architecture knowledge

  Software architecture design knowledge can be characterized by information that supports software architecture processes.
- Architecture knowledge management

  Software architecture knowledge management is an approach to improving software architecture process outcomes by introducing practices for identifying and capturing architecture knowledge and expertise, and making it available for reuse across projects.

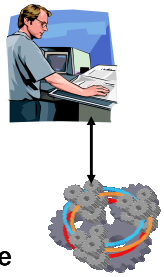
Architecture Knowledge Management Issues

- Unavailability of architecture design knowledge
  - System evolution becomes hard
  - Difficult to identify design errors
- Use of COTS without fully understanding the assumptions
- Situation is worst in case of FLOSS components
- Lack of support in terms of practices and tools
  - What types of architecture knowledge are useful?
  - How to store and manage the knowledge?
  - Make knowledge capture cost-effective

Categories of Factors

- Organization
  - Leadership
  - Processes
  - Platform
  - Structure/Culture
- Human/Behavioral
  - Training
  - Common language
  - Culture
  - Motivation
- Technological
  - KMSS
  - Communication
  - Collaboration
  - Coordination

Knowledge Management Strategies

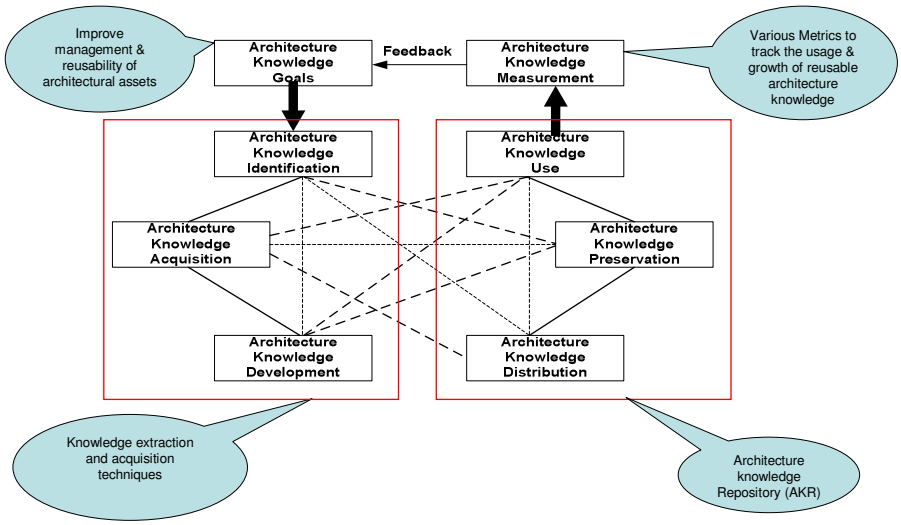


- Codification
  - High quality, reliable, and rapid use of knowledge
  - High IT investment
  - Rewards for sharing & reusing codified knowledge
- Personalization
  - Connecting people & channelling individual expertise
  - Moderate IT investment
  - Reward for sharing knowledge face to face
- Hybrid approach
  - Codify organization-wide knowledge
  - Personalize local & context specific knowledge

Knowledge Management (KM) Task Model

- KM task model consists of eight tasks
- Strategic tasks
  - Architecture knowledge goals
  - Architecture knowledge measurement
- Operational tasks
  - Architecture knowledge identification
  - Architecture knowledge acquisition
  - Architecture knowledge development
  - Architecture knowledge distribution
  - Architecture knowledge use
  - Architecture knowledge preservation

Supporting Knowledge Management Tasks



Metrics for Knowledge Management

Some of the measures that can be used for most of the KM initiatives

Outcome

- Time, money or personal time saved
- Percentage of successful project compared to those before KM
- Time and number of employees trained

Output

- Usefulness surveys where users evaluate how useful KM initiative have been in helping them accomplish their objective
- Usage anecdotes where users describe how the initiative has contribute to their daily tasks and overall efficiency and effective of organisation

System

- Dwell time per page or section
- Latency (response time)
- Number of downloads
- Number of site access
- Usability survey
- Frequency of use
- Navigation path analysis
- Number of helpdesk calls
- Number of users
- Frequency of use and/or update
- Percentage of total employees using system

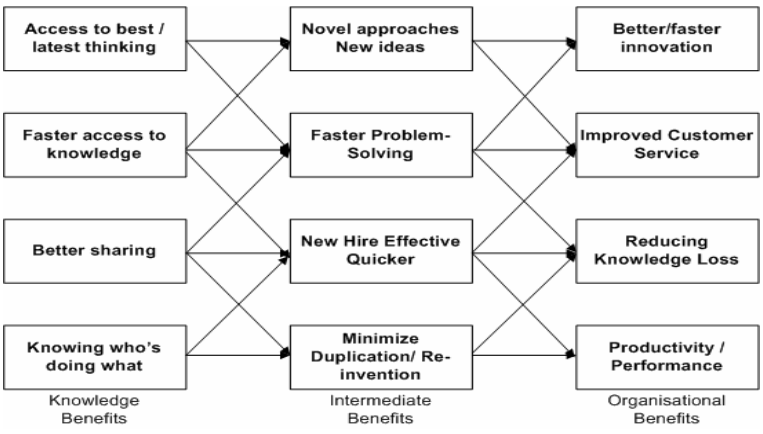
Benefits of Architecture Knowledge Management

lero THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

- Expected benefits:
  - Capture rationale for architecture decisions
  - Help build architectural capabilities
  - Improve architectural reusability
  - Support better quality decisions
  - Minimize architecture risks
  - Minimize reoccurrence of design mistakes
  - Avoid dependency on key individuals
  - Gain competitive advantage
  - Encourage best architectural practices
  - Improve efficiency of architectural processes
  - Support case-based reasoning

Knowledge Management Benefits Tree

lero THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE



Source: David Skyrme associates

A Few Useful Tools

- Protégé
  - Creation & manipulation of Ontologies for domain modeling
  - Knowledge acquisition & management
- Chimaera
  - Creating & maintaining distributed Ontologies
  - Support for loading, browsing, reorganizing Ontologies
- Swoop
  - Hypermedia inspired Ontology editor
- BRedB
  - Support for capturing and contextualizing knowledge
  - Knowledge management and reuse

R&D in Architecture Knowledge

- Grady Booch – Handbook of software architecture
- Philippe Kruchten – Design decision ontology and use case model for tool support
- GRIFFIN Project – Architecture knowledge modeling, representation, and acquisition approaches
- Anton Jansen & Jan Bosch – Describing architecture as design decisions in Archium
- Antony Tang & Jun Han – Architecture rationale and elements linkage (AREL) for capturing and traversing rationale to support design decision reasoning

A Preview of our R&D

- Repository of architecture knowledge
  - templates for capturing design elements and diagrams
- Tool support for design decision Ontology and Use Cases
- Architecture knowledge model, Use case model, and knowledge acquisition approaches
- Describing architecture as design decisions using templates – different levels of abstractions
- Our approach complements AREL's UML based rationale with templates and search mechanism

Case Study Description

- Organizational context
- Domain description
- Software architecture evaluation challenges
- Software architecture knowledge management initiative
- Logistics
- Progress so far!

Summary of Section 1

- What is architecture knowledge?
- What is architecture knowledge management?
- Why is architecture knowledge management important?
  - Expected benefits
  - Challenges
  - Strategies
- State-of-the art
  - Tooling support
  - R&D
- Case study description

A Tool and Approaches for Managing Knowledge

- An Introduction to BRedB – An Architecture Knowledge Management Infrastructure
  - Architecture
  - Features to support architecture design, documentation, and evaluation activities
- Capturing software architecture knowledge
  - Approaches
  - Advantages and disadvantage
- Using BRedB on the case study

Tool for Managing Architecture Knowledge

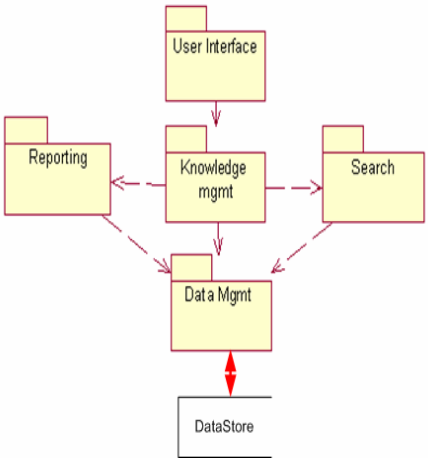
lero THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

- BRedB: A Process-based Software Architecture Knowledge Management Infrastructure
  - A model of architecture knowledge management
  - Templates for capturing and structuring architecture knowledge
  - Repository of architectural knowledge and experiences
  - Resource for software architects
  - Source of adaptive software architecture processes
  - Support for owning technical and process knowledge

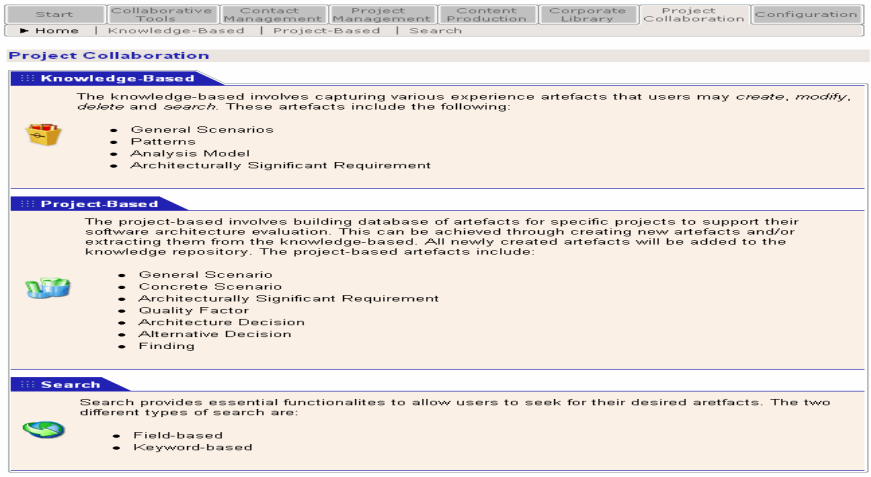
Component View of PAKME Architecture

lero THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

- User Interface
- Knowledge Management
- Knowledge Search
- Reporting
- Data Management
- Knowledge Repositories



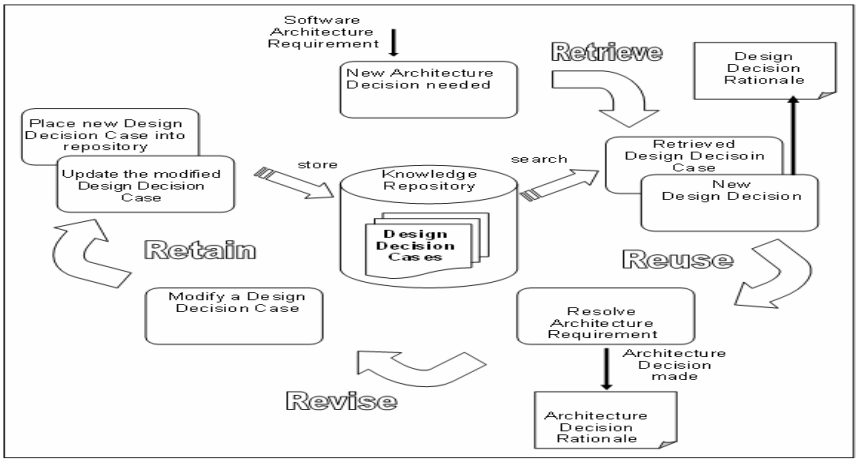
GUI for Start Using BRedB



Support for Architecture Design

- Support for case-based reasoning by human expert
- Repository of reusable architectural artifacts
- Capture/access rationale for design decisions
- Catalogue of architecture and design patterns/tactics
- Search architectural artifacts and knowledge

Capturing and Reusing Cases of Design Decisions

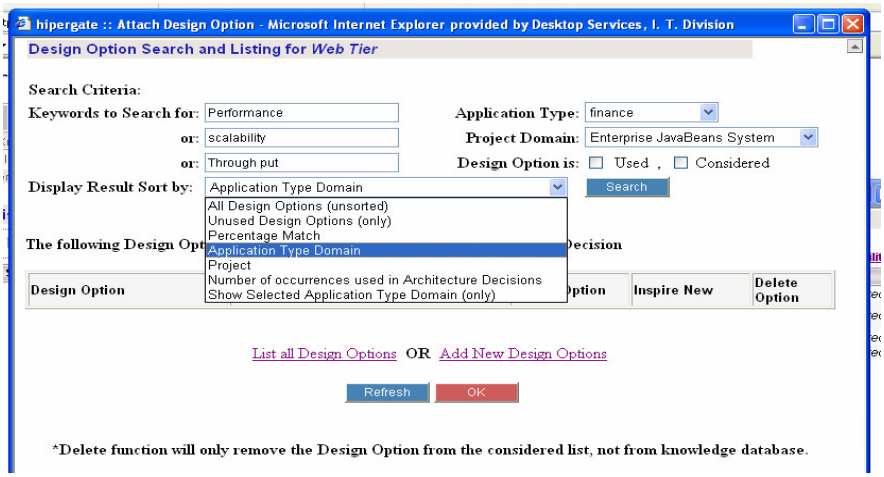


Design Decision Captured as a Case

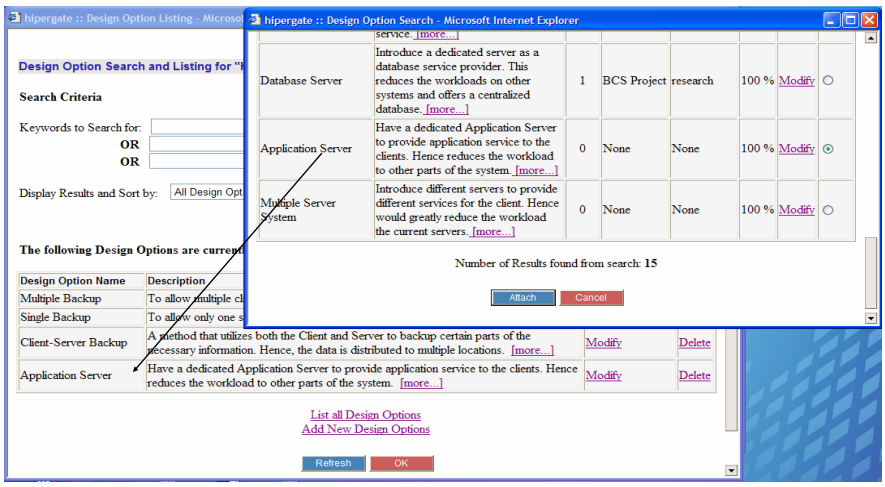
hipergate :: View Design Option - Database Server - Microsoft Internet Explorer

Rationale	View Design Option Rationale			
Used in Architecture Decision	**Architecture Name**	**Description**	**Project Name**	**Project Domain**
	High Server Performance	Require fast response times from the server. [more...]	BCS Project	research
Inspiration	This design, "Database Server", was inspired by the following Design Options: [Find more Inspiration...]			
	Design Option	**Description**		
	Secondary Server System	A Secondary Server is installed onto the system to help share the workload. Not only will this help improve the efficiency, but if the primary server failed, then the secondary server can continue the service. [more...]		
	Backup Server System	Introduce an extra server as backup. The extra server will be connected into the system but will only run when the primary server has failed. Hence users would not feel a lost in service. [more...]		
Inspired other Design Options	This design, "Database Server", inspired the following Design Options:			
	Design Option	**Description**		
	Multiple Server System	Introduce different servers to provide different services for the client. Hence would greatly reduce the workload the current servers. [more...]		
Modify	Modify current Design Option			

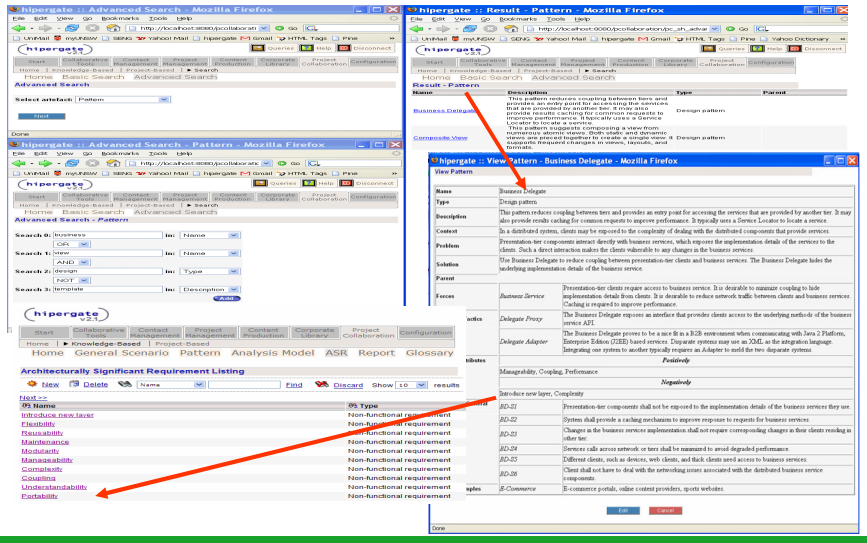
Searching Design Decision Cases



Using a Design Decision Case



Navigating the Knowledge Base



Template for Capturing and Representing Patterns

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

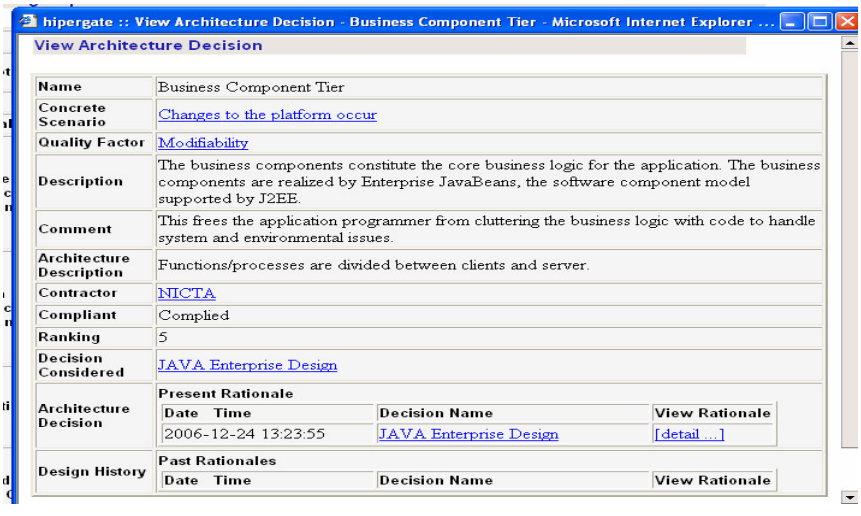
Support for Architecture Documentation

lero THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

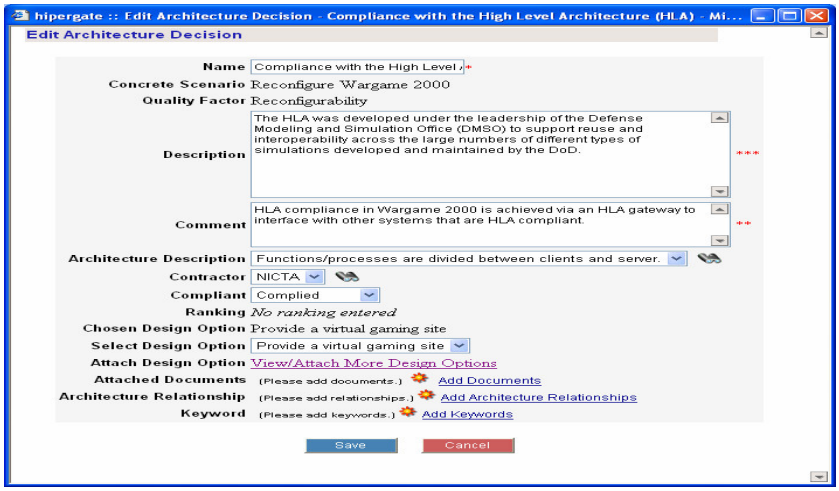
- Templates for documenting design decisions
- Store architectural models and documents
- Support for standards such as IEEE 1471-2000
- Represent architectural decisions using views
- Attach process knowledge to architectural artifacts

Structuring and Representing Architecture Decisions

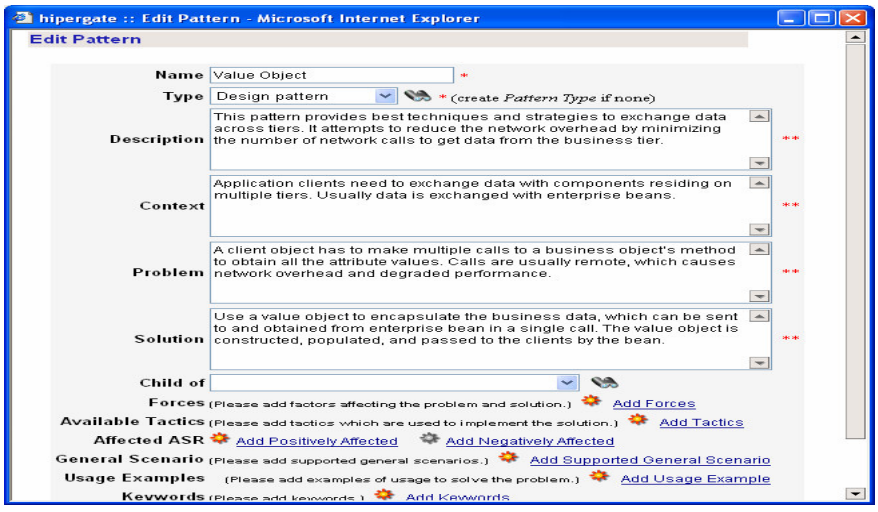
lero THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE



Modifying an Architecture Decisions



Modifying a Pattern



Support for Architecture Evaluation

THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

- Repository of general scenario to support QAWs
- Capture rationale and contextual information surrounding design decisions
- Search and view rationale for previous design decisions
- Documenting findings of evaluating design decisions
- Categorize findings in suitable risk themes
- Generate evaluation reports for the management

General Scenario Captured in BRedB

THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

General Scenario Listing

New Delete Accept Reject Name Search Discard Show 50 results

Proposed General Scenarios

<< Previous

Name	Description	Source	Date Entered	Logs
SEC-1	QVS shall accept online payments for the services that means the transactions between the QVS and financial institutes must be protected.	User-Defined	Sun 24 Dec 2006 19:47	
SEC-2	QVS provides secured storage to customers' credit details and other information.	User-Defined	Sun 24 Dec 2006 19:47	
SEC-3	QVS shall be able to identify different users and verify their access privileges according to their memberships of different user groups.	User-Defined	Sun 24 Dec 2006 19:49	
SEC-4	QVS shall be able to detect and prevent Denial Of Service (DOS) attacks. The system shall be able to run reliably most of the time.	User-Defined	Sun 24 Dec 2006 19:49	
SEC-5	QVS is an evolving system that shall be easily modifiable to introduce changes in the security policy and other security checks.	User-Defined	Sun 24 Dec 2006 19:50	

Accepted General Scenarios

<< Previous

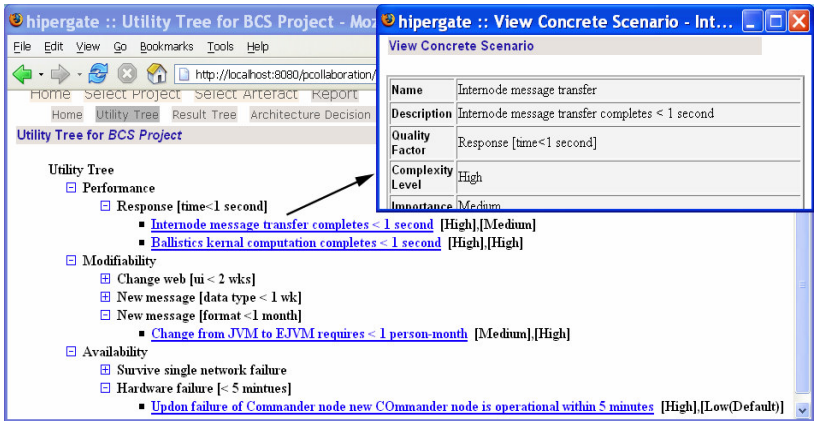
Name	Description	Source	Date Entered	Logs
Performance	Require bounded response time and/or certain system throughput.	User-Defined	Tue 19 Dec 2006 11:28	
Changing the hardware platform	Change physical location of service with minimal impact on the rest of the system.	Pattern	Tue 19 Dec 2006 10:23	
Changes number of users	Number of users changes while maintaining other qualities such as performance.	Pattern	Tue 19 Dec 2006 10:50	
Change the implementation	Implementation details change without affecting much of the rest of the system.	Pattern	Tue 19 Dec 2006 11:49	
Addition of functionality	Addition of functionality without impacting the rest of the system.	Pattern	Tue 19 Dec 2006 14:49	
Availability	An internal or external component fails and the system is able to recognize the failure and has strategies to compensate for the fault.	Pattern	Tue 19 Dec 2006 12:04	
Not prepared event	An event arrives at the system for which it was not prepared.	User-Defined	Tue 19 Dec 2006 15:20	
0	0	User-Defined	Thu 21 Dec 2006 10:12	

Rejected General Scenarios

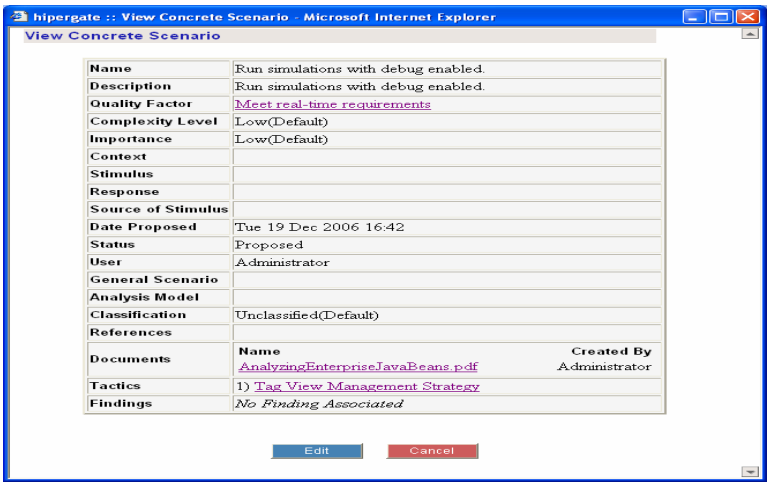
<< Previous

Name	Description	Source	Date Entered	Logs
BD-S3	Changes in the business services implementation shall not require corresponding changes in their clients residing in other tier.	User-Defined	Fri 28 Jan 2005 11:02	
BD-S1	Presentation-tier components shall not be exposed to the implementation details of the business services they use.	Pattern	Fri 28 Jan 2005 11:01	
BD-S5	Different clients, such as devices, web clients, and thick clients need access to business services.	Pattern	Fri 28 Jan 2005 11:16	
BD-S4	Services calls across network or tiers shall be minimized to avoid degraded performance.	Pattern	Fri 28 Jan 2005 11:02	

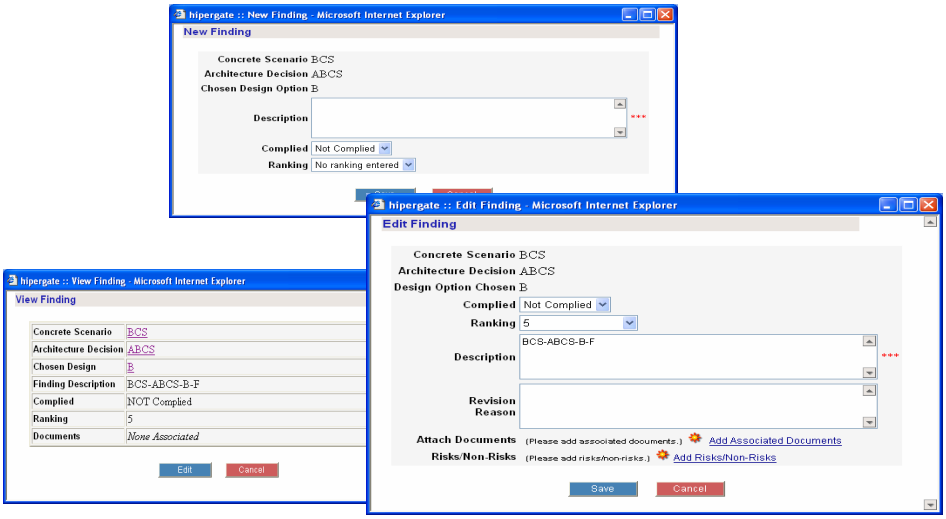
Utility Tree of Concrete Scenarios



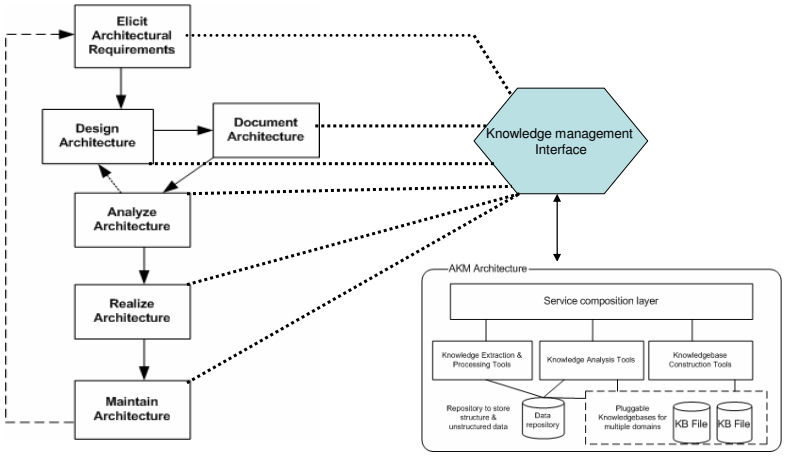
Viewing Details about a Concrete Scenario



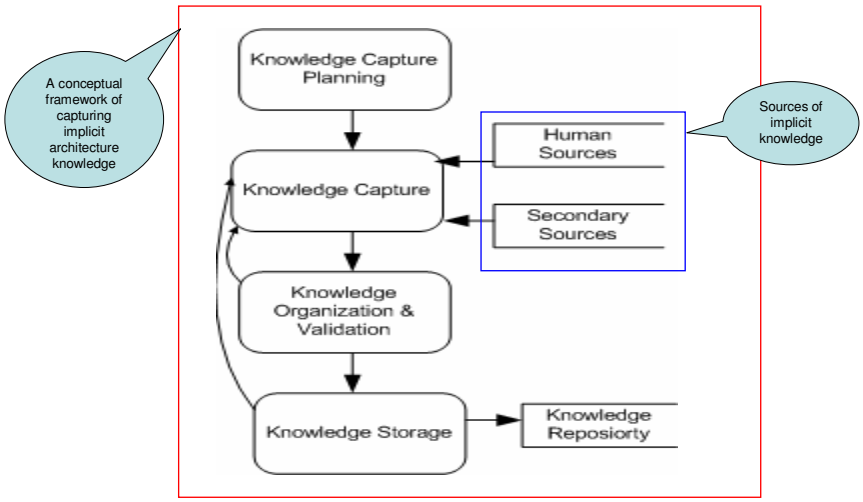
Documenting and Viewing Findings



Support for Architecture-Centric Development



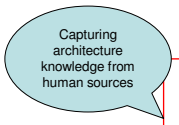
Capturing Architecture Knowledge



Source of Architecture Knowledge

- Human sources
  - Stakeholders, BA, architects, developers
- Secondary sources
  - Design patterns, books, case studies

Knowledge Acquisition from Human Sources



Individual Knowledge Acquisition Techniques	Team Knowledge Acquisition Techniques
Interviewing Questionnaire Observation Protocol analysis Repertory grid analysis	Brainstorming Architecture reviews Focus group interviews Delphi technique Group repertory grid analysis Group support systems

Individual Knowledge Acquisition Approaches

- Interviewing
- Questionnaires
- Observations
- Prototype analysis
- Repertory grid analysis

Team Knowledge Acquisition Approaches

- Brainstorming
- Nominal group technique (NGT)
- Focus group interviews
- Delphi technique
- Group repertory grid analysis
- Architecture reviews

Disadvantages

- Individual Knowledge Acquisition Approaches
  - Time consuming
  - Subjective opinion or interpretation
  - Need considerable domain knowledge
  - May result in fabricated information because of defensive attitude
  - Inconsistent knowledge because of integration issue
- Team Knowledge Acquisition Approaches
  - Expensive activities with multiple logistical issues
  - Require an expert moderator
  - May cause non-participation and conflicts

A Comparative View of Other Approaches

Method category	Description	Advantages	Disadvantages
Method of familiar tasks	Analysis of the tasks that the expert usually performs.	The expert feels comfortable	Can be fairly time-consuming
Structured and unstructured interviews	The expert is queried with regard to knowledge of facts and procedures.	For a first-and second pass at a data base, it can generate much information.	Typically very time consuming
Limited information tasks	A familiar task is performed, but the expert is not given certain information that is typically available.	Can be tailored to extract information on selected sub-domains of knowledge	Expert feels uncomfortable and is hesitant to make judgements
Constrained processing tasks	A familiar task is performed, but the expert must do so under time or other constraints.	Can be tailored to extract information on selected sub-domains of knowledge, or on the expert's strategies	Expert feels uncomfortable and is hesitant to make judgements.
Method of "tough cases"	Analysis of a familiar task that is conducted for a set of data that presents a "tough case" for the expert.	Can yield information about refined reasoning	Occur unpredictably, the knowledge acquirer may not be present

Source: Experimental Psychology by Robert R. Hoffman, AI Magazine, summer 1987.

Pattern-Based Architecture Knowledge

- Exploiting relationships among scenarios, quality attributes, and patterns as architecture knowledge
- Patterns widely codified
  - Links among scenarios, quality attributes and patterns
- We have observed that patterns
  - Include implicit scenarios and links between quality attributes and patterns
  - Provide a previously untapped source of reusable architecturally significant information
- Extracting the information
  - Relatively time-consuming and required expertise

Mining Patterns for Architecture Knowledge

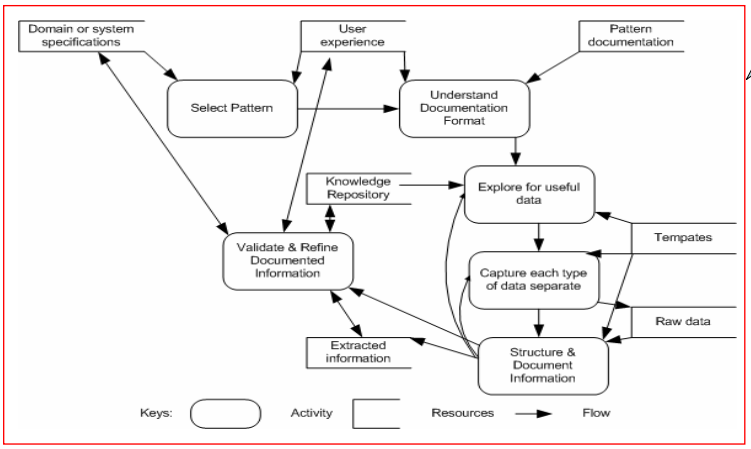
lero THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

- A mechanism for extracting (*mining*) architecture data from patterns
  - General process model
    - Steps involved in mining patterns with guidelines
  - Templates to record
    - Generic architecture information
    - Project related information relating to concrete scenarios
- Intended to reduce the time and expertise needed to extract architecturally significant information from patterns

Pattern-Mining Process Model

lero THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

Process model for mining patterns for architecture knowledge



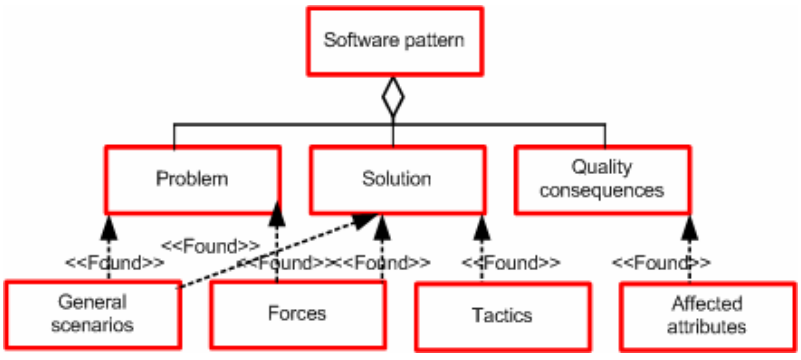
Pattern-Mining Process

lero THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

- Select software pattern to be explored for architectural information
- Understand the pattern documentation format
- Identify architectural information described in a pattern's documentation
- Capture each type of information separately
- Organise the extracted information using the provided template
- Validate and refine documented information

Diagrammatic Guidelines for Mining Pattern

lero THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE



Documenting the Extracted Information

Pattern Name: *Name of the pattern*		**Pattern Type:** *Architecture, design, or style*	
Description	*A brief description of the pattern.*		
Context	*The situation for which the pattern is recommended.*		
Problem	*What types of problem the pattern is supposed to address?*		
Suggested	*What is the solution suggested by the pattern to address the problem?*		
Forces	*Factors affecting the problem and solution and pattern's justification.*		
Tactics	*What tactics are used by the pattern to implement the solution?*		
Affected Attributes	**Positively**	**Negatively**	
	Attributes supported	*Attributes hindered*	
General scenarios	**S**	*A textual, system independent specification of a quality attribute.*	
	S		
Example	*Some known examples of the usage of the pattern to solve the problems.*		

Using the Template

Pattern Name: *Business Delegate*		**Pattern Type:** *Design pattern*	
Brief description	*This pattern reduces coupling between tiers by providing an entry point for accessing the services another tier. It also supports results caching to improve performance…*		
Context	*A client may be exposed to the complexity of dealing with the distributed components…*		
Problem description	*Presentation-tier components interact directly with business services. Such a direct interaction makes the clients vulnerable to any changes in the business services…*		
Suggested solution	*Reduce coupling between presentation-tier clients and business services. The Business Delegate hides the underlying implementation details of the business service...*		
Forces	*Presentation-tier clients require access to business service.* *It is desirable to minimize coupling to hide implementation details from clients.*		
Available tactics	*Delegate Proxy and Delegate Adapter*		
Affected Attributes	**Positively**	**Negatively**	
	Reduce coupling, manageability, performance	*Introduce new layer, increased complexity*	
General scenarios	**S1**	*Presentation-tier components shall not be exposed to the implementation details of the business services they use.*	
	S2	*System shall provide a caching mechanism to improve response to business service request.*	
	S3	*Services calls across network or tiers shall be minimized to avoid degraded performance.*	
Examples	*E-commerce portals, online content providers, sports websites.*		

A Template for Documenting Concrete Knowledge

Project Name: *Which project needs this scenario?* **Project domain:** *Domain of the project*		**Date:** *When was proposed?* **Scenarios No:** *Serial number assigned to the scenario*
Business goals	*Which business goals does this scenario achieve?*	
Stakeholders	*Which class of the stakeholders did suggest this scenario?*	
Attributes	*Which quality attributes are required by this scenario?*	
Description	*A brief description of the scenario.*	
Concrete scenario	**Stimulus**	*A condition that needs to be considered when it arrives at a system.*
	Context	*A system's condition when a stimulus occurs, e.g. overloaded, running etc.*
	Response	*A measurable action that needs to be undertaken after the arrival of the stimulus*
	Complexity	*How complex is this scenario to realize? (Effect on macro or micro architecture)*
	Priority	*How important is this scenario?*
Pattern/Style	*Name of the architectural pattern or style that can support this scenario.*	
Design tactics	*What are the design tactics used by the pattern/style to support the scenarios?*	
Design rational	*What are reasons for using the patterns/tactics? How does it provide the desired quality attributes?*	

Documenting Concrete Knowledge Example

Project Name: *Qualification Verification System* **Project domain:** *E-Commerce application*		**Date:** *12/06/2005* **Scenarios No:** *Serial number assigned to the scenario*
Business goals	*Customer satisfaction and process efficiency.*	
Stakeholders	*Business Manager, System sponsors, and End User.*	
Attributes	*Improved performance*	
Description	*The response to a business service request shall be improved to avoid users' frustration and system shall be able to handle up to 1000 users concurrently without any delay in the response time.*	
Concrete scenario	**Stimulus**	*A user request needs to be processed.*
	Context	*There are 1000 users, who may request for a service simultaneously.*
	Response	*The system shall be able to respond to a request within 2 seconds.*
	Complexity	*Medium*
	Priority	*High*
Pattern/Style	*Business Delete*	
Design tactics	*Delegate proxy and Caching*	
Design rational	*This pattern exposes an interface to the business service API by using proxy function to pass the client methods to the session bean. It can cache any necessary data and references to the session bean's home or remote objects to improve performance by reducing the number of lookups.*	

Value of Mining Pattern

- During architecture design
  - Helps identify patterns used in a software architecture and understand the reasons for using those patterns.
  - Helps identify appropriate patterns for system design.
- During architecture analysis
  - Provide confidence in an architecture's capability of supporting certain scenarios
  - Help develop concrete scenarios based on general scenarios extracted from the patterns in the architecture

Empirical Evidence

- Research Program
  - Observational study and controlled experiments
- Pattern-Mining Framework
  - Framework is effective and it takes 30-45 minutes to mine one pattern
- Understanding and designing architecture
  - ASIP compared with standard pattern documentation is more helpful in understanding and designing architectures
- Architecture evaluation
  - Improves scenario gathering activity by helping stakeholders to develop better quality scenarios

Trial of the Technology

- Tailoring BRedB for DSTO
  - Define domain-specific evaluation criteria
  - Capture knowledge underpinning COTS' architectures
  - Support standards compliance analysis
  - Document rationale for COTS acquisition decisions
  - Rank and compare architectural solutions
  - Track architecture decision makers and evaluators
  - Generate reports for management decision making

Summary of Section 2

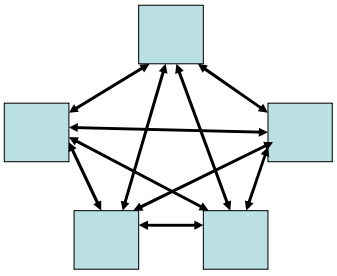
- An Introduction to BRedB – An Architecture Knowledge Management Infrastructure
  - Architecture
  - Features to support architecture design, documentation, and evaluation activities
- Approaches to capturing architectural knowledge
- Using BRedB on the case study

Domain Modeling for Architecture Knowledge

- Knowledge modeling
- Knowledge modeling process
- Knowledge modeling techniques
- Architecture knowledge constructs & relationships
- Models characterizing architecture knowledge
- Some example queries for case study

Knowledge Modeling

- Why do we need models?
- Importance of knowledge modeling
- Role of models in knowledge acquisition and validation
- Modeling in software engineering and knowledge engineering: similarities and differences
- How to model knowledge?



A Generic Process of Modeling Knowledge

- Domain analysis for identifying constructs and their attributes
- Identify relationships among different constructs
- Determine properties of relationships
- Apply some formal modeling approach, e.g. Entity-Relation models
- Refine the conceptual model using the principles of a formal modeling approach
  - Avoid fully normalizing the model
- Assessment – many ways such as expert opinion or theoretical

Knowledge Modeling Techniques

- CommonKADS
- Protégé
- UML & OCL
- Multi-perspective modeling

Comparison of Knowledge Modeling Techniques

Technique / Features	CommonKADS	Protégé 2000	Multi-Perspective	UML
K.E. methodology	X			
O-O approach	X	X	X	X
Platform independent	X	X	X	X
Hybrid approach	X		X	X
Editor tool		X		
Standard modeling language				X
Documentation	X	X	X	X
Evolving		X	X	X
Domain	Medical, Business, & Legal	Medical, Business, & Legal	Medical, Business, & Legal	Medical, Business, & Legal
Other features (OKBC, RDF, Semantic Web)		X		

Source: M.S. Abdullah et al. 3rd European Conf. on KM, 2002

Exercise – Domain modelling

- Form a group of 2-3 and identify constructs, their properties, and relationships that characterise architecture knowledge in your domain/industry.

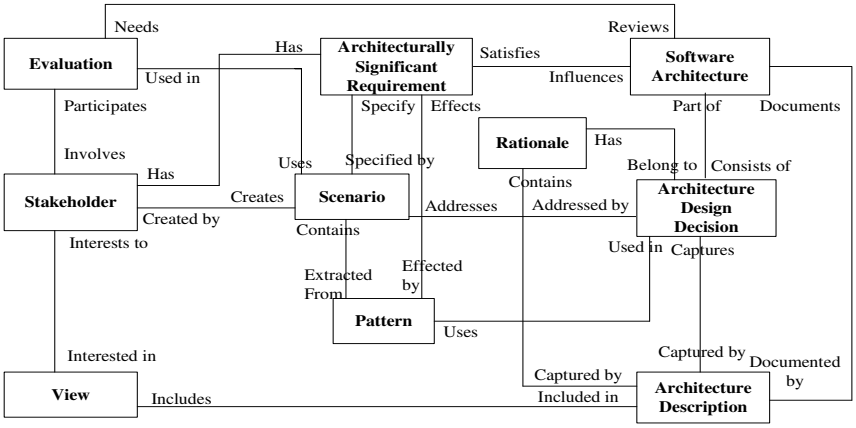
Architectural Knowledge – Some Constructs

- Architecture decision
- Design option
- Design rationale
- Pattern – (Architectural, design, platform)
- Tactic
- Architectural significant requirement (ASRs)
- Scenarios
  - Abstract
  - General

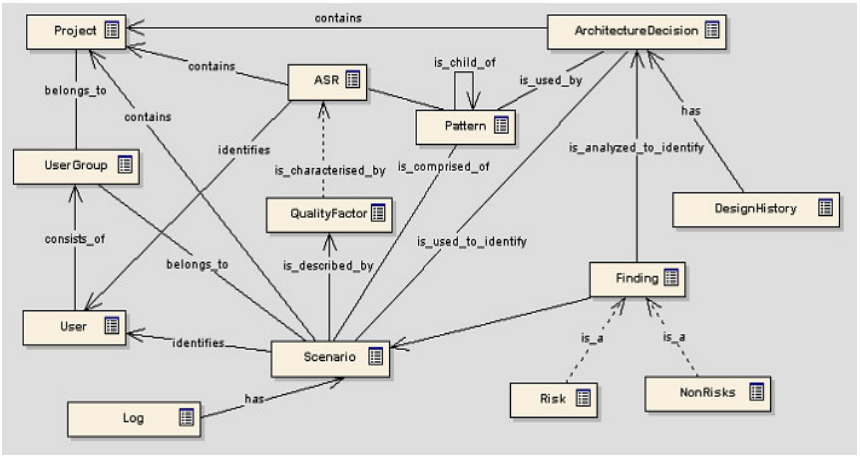
Some of the Sources for Identifying Constructs

- SARA report
- IEEE 1471 standards
- Views and Beyond
- Books on software architectures
- Literature on software architecture

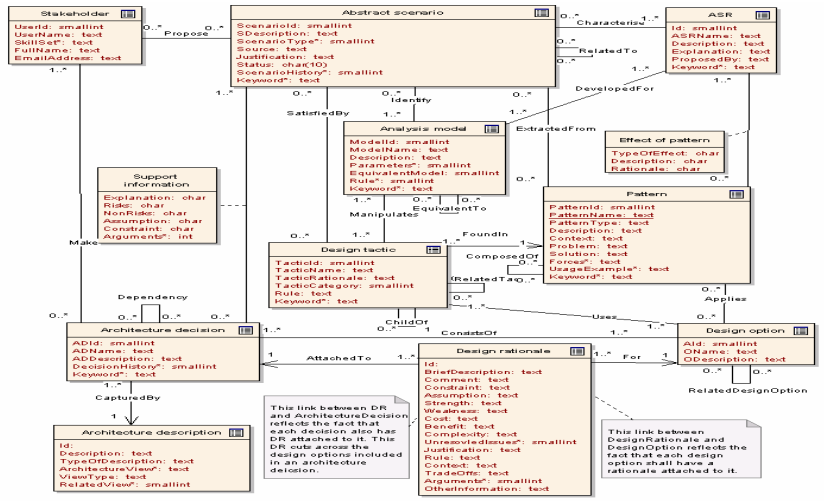
A Schema for Architecture Knowledge



A partial model of Architecture Knowledge



Characterising Architecture Knowledge



A Few Heuristics for Modeling Knowledge

- identify most-frequent, synonyms or related terms
- Identify relationships between terms and models
- Identify basic constraints
  - Pre-requisite constraint
  - Temporal constraint
  - Mutually inclusive constraint
  - Mutually exclusive constraint
- Identify domain constraints
- Identify domain dependencies

Summary of Section 3

- Knowledge modeling
- Knowledge modeling techniques
- Architecture knowledge modeling process
- Architecture knowledge constructs & relationships
- Models characterizing architecture knowledge
- Some example queries for case study

Using knowledge during design & evaluation

- Industrial case study – Avionics Architectures
  - Problem domain description
  - Current evaluation methods
- A framework for avionic architecture evaluation using knowledge management
  - Quality attributes
  - Evaluation framework
  - Risk management
- Demonstration: Using BRedB for avionics architecture evaluation

Airborne Mission Systems (AMS) Branch

- Research, analytical studies and experimental work in Avionics and AMS.
- Development of innovative methods and tools to aid the understanding of advanced avionics technologies
- Provide short-term advice to Australian Defence Force (ADF) project teams and develop long-term strategies to support the ADF on mission systems acquisitions.

Airborne Mission Systems (AMS) Branch

- Responsible for performing Technical Risk Assessments (TRA) of mission systems
  - critical for the ADF acquisition process
- AMS is required to understand and organize large amounts of architecture design knowledge
- "Software intensive projects are historically considered the most risk prone in the Defence domain"
- Building capabilities in systematically evaluating architectures and maintaining architecture knowledge

ADF Acquisitions and Software related Issues

- Navy's $100m chopper can't fly in bad light
  - $1.1b were spent before finding that Seasprite had major software problems
- Billions wasted on 'Collin Class Submarines'
  - Software and architectural issues made them 'Duds'
- Airborne Early Warning and Control project has 4 million lines of code
- Software literacy is vital but quite low
- Poor record keeping makes it impossible to trace reasons for bungles in acquisition projects

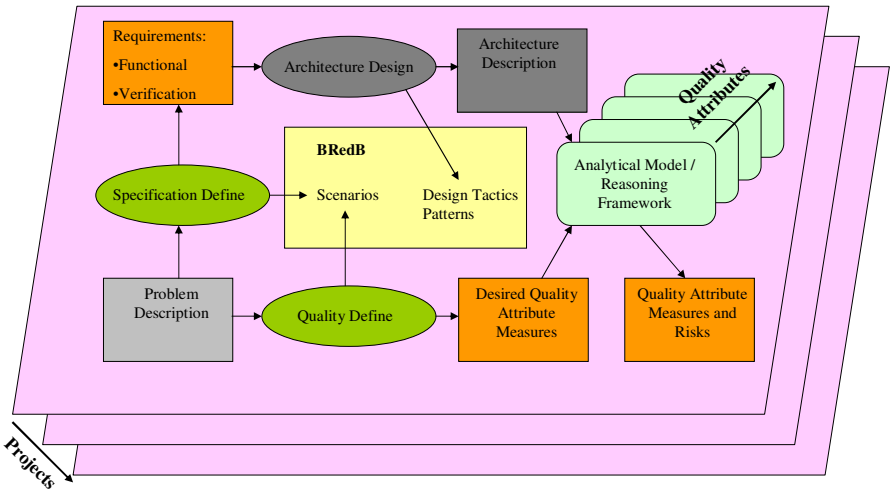
Trial's Background

- Collaborative R&D project between NICTA and DSTO has been undertaken during 2006
- Exploit NICTA architecture evaluation methods and tools for improving AMS capabilities
- Codifying architecture evaluation process and design knowledge and rationale is a vital goal of this project
- 8 members team (4 on the each side)

Trial's Objectives

- Support a systematic and repeatable evaluation process
- BRedB is expected to help AMS to achieve several goals:
  - Capture rationale for architecture decisions
  - Help build architectural capabilities
  - Improve architectural reusability
  - Provide an audit trail for TRA findings
  - Reduce demands on subject matter experts
  - Encourage best architectural practices
  - Improve efficiency of architectural processes
  - Accelerate the training process of new employees

BRedB Supported Architecture Evaluation Process



Tailoring BRedB 1/3

- AMS doesn't design - it evaluates architectures proposed by contractors in response to a Request for Proposal (RFP)
- AMS only needs BredB features that support architecture evaluation activities
- Gathering customization requirements by analysing activities/tasks of the current process

Tailoring BRedB 2/2

- High priority requirements implemented:
  - Classification of project data according to the Defence classification scheme.
  - Mechanism for recording compliance of architectural decisions with respect to requirements.
  - Store and evaluate several contractors' proposals for the same set of scenarios within one project.
  - Different levels of access to project data based on the Defence security scheme.
  - Ability to import/export data from the tool based on the classification scheme.
  - Integration with requirements management and architecture modelling tools.

AMS Case Study

- Assessing customized BRedB by supporting architecture evaluation process of AMS
- Evaluating the proposed architecture for an airborne mission system
- Goal was to investigate BRedB's role in the architecture evaluation process and how it could help capture and manage architecture knowledge and rationale

Case Study – Use of the Knowledge Base

- BRedB was populated with AMS domain knowledge, starting with a preliminary domain specific quality model.
- Quality model was based on ISO 9126 and AMS domain knowledge elicited from domain experts
- Quality model is used to assess the potential risks of architectural designs
- Quality model consists of six quality attributes:
  - Performance
  - Reliability
  - Usability
  - Maintainability
  - Portability
  - Functionality

Quality Framework - Performance

THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

Latency	What are average/worst case latencies for behavior such as: Radar to display, Algorithm used to calculate tracks/bearings, and Video frame updates.
Accuracy	Accuracy relates to timestamping of events. Check the architecture for: place of time reference, multiple clocks are maintained/synchronized, clock signal distribution through aircraft?
Capacity	Look for the following data related to capacity: Number of tracks a system can support/process and database size/storage, and processor capacity.
Throughput	Check if the proposed processor and bus architecture can support the likely throughput in terms of events per second (both average and worst case)
Resource usage	Look to assess several things in terms of resource usage: CPU (peak CPU load, spare capacity), Power consumption, Cooling, Memory, Disk (usage for each disk), and I-O rates.
scalability	Various questions to be asked such as Are any key software elements single-threaded? Do the specifications indicate spare CPU capacity that could be used for additional processing?
Schedulability	Is it likely that high priority taks can be scheduled when needed? What is the clock granularity

Scenarios – Performance

THE IRISH SOFTWARE ENGINEERING RESEARCH CENTRE

Initialization	Must perform all initialization activities within 10 minutes.
Latency	Run simulations with no instantaneous lags greater than five seconds, no average lags greater than three seconds.
Capacity	Run-time simulation with debug enabled.
Latency	Finish data collection within 30 seconds of simulation termination.
Throughput	Finish data collection request from three network sensors within x seconds every one minute.

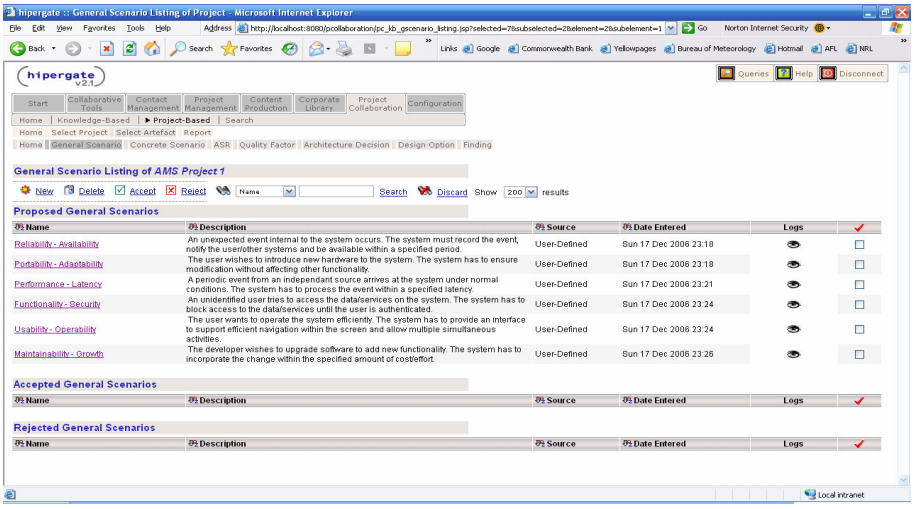
Quality Framework – Reliability

Fault tolerance	Look for a fault tolerance in terms of: software and hardware redundancy, separation of flight /mission functions, and potential single points of failure, mechanics of notify operators of faults.
Recoverability	Check for time required for cold/warm start ups, for cold start to full performance, amount of data loss on a failure/ restart, any loss of situational information and how to recover/readjust to that.
Maturity	Are existing tested components used? Level of process maturity of the organization.
Survivability	Check for the kind of redundancy for key software/hardware components, communication channels.
Availability	Check for restart/recovery times after failure and supportability.

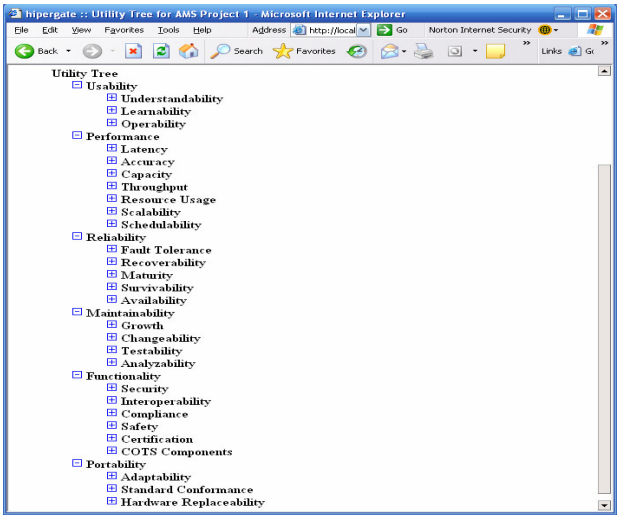
Scenarios – Reliability

Availability	Simulation controller initiates simulation execution (a game), starts subordinate processes, loads parameter files, and simulation runs from T=-infinity to T=0 within 10 minutes.
Availability	Runs scenarios without crashes: no hardware and/or software failures. Terminates normally.
Fault tolerance	Ability to change the location and orientation of assets and still meet the performance, reliability, and credibility criteria.
Maturity	Organization has been developing similar systems for the last 20 years and has been certified as CMMI level 4.
Recoverability	System encounters serious data corruption and needs to restart. System should recover in 30 seconds.

Quality Framework in BRedB

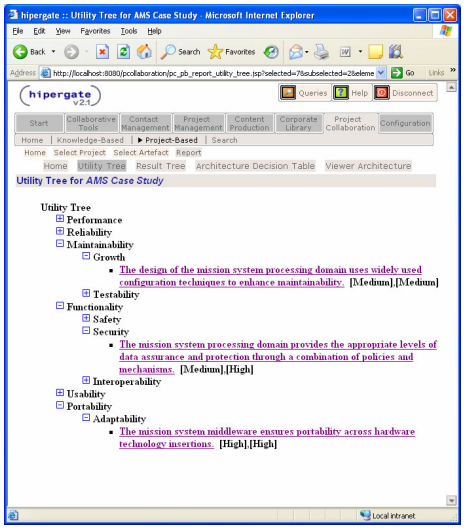


Utility Tree of the Quality Framework



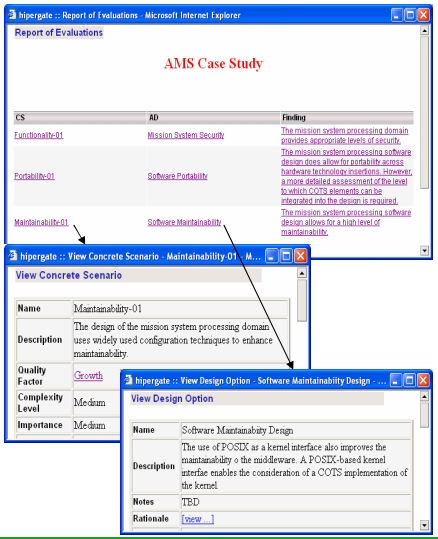
Case Study – Use of the Project Base

- Concrete scenarios relating to the quality factors were added to the Project Base.
- The architecture design was captured and compared with alternative designs with respect to the scenarios



Case Study – Reporting

- A report of the evaluation shows the findings of an architectural decision that is aimed at satisfying a concrete scenario



Challenges

- Domain understanding
- Access to the required information/knowledge
- Security modelling and implementation
- Modification of the underlying data model
- Communication/coordination issues

Initial Findings

- The use of an evaluation framework and knowledge management tool brings added rigour to the evaluation process.
- It is anticipated that access to past project experience stored in BRedB will be valuable in the evaluation process.
- The modified version of BRedB provides AMS with an effective and efficient mechanism to organise and understand large amounts of architecture knowledge

Tool Demo

- Short Video clips on using BRedB and Wiki based tool

Summary of Section 4

- Industrial case study – Avionics Architecture
- A framework for avionic architecture evaluation using knowledge management
  - Quality attributes
  - Evaluation framework
  - Risk management
- Demonstration: Using BRedB for avionics architecture evaluation

Some useful references

- Bass et al. – Software Architecture in Practice
- Clements et al. - Documenting Software Architectures : Views and Beyond
- Ian Gorton – Essential Software Architecture
- IEEE Recommended Practice for Architecture Description of Software-Intensive System (IEEE Std 1471-2000)
- Philippe Kruchten - http://philippe.kruchten.com/
- Grady Booch's site

Acknowledgement

- Tutorial incorporates material provided by Andrew Northway, Paul Heuer, and Thong Nguyen of DSTO. Discussions with our collaborators Antony Tang and Jun Han resulted in many ideas presented in the tutorial.
- Shirley Xu worked on the engineering side of the BRedB. Before that several students of the UNSW helped us build the first version of that tool.
- Most of the reported work was produced while the presenters were working at NICTA

Collaborative Opportunities

- We welcome collaborative opportunities to deploy our tools and techniques to help design better architectures and effectively and efficiently evaluate and manage them.

Contacts:

- Ali Babar, Lero, Ireland (malibaba@lero.ie)
- Ian Gorton, PNNL, USA (ian.gorton@pnl.gov)

Thank you!