Agility and Architecture: Why and How They can Coexist?

M. Ali Babar

IT University of Copenhagen, Denmark

Keynote, Third Turkish Software Architecture Conference

Ankara, Turkey, November 4, 2010

Background Brief

M. Ali Babar

Associate Professor @ ITU

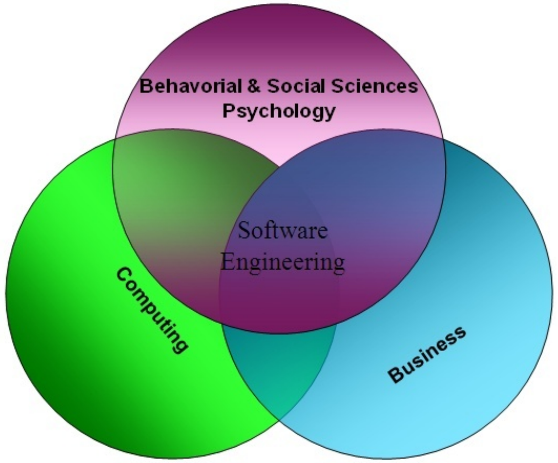
PhD in CSE, University of New South Wales

Work History:

- ITU, CPH: 2009 …
- Lero, Ireland: 2007 – 2009
- NICTA, Australia: 2003 - 2007
- JRCASE, Macquarie University: 2001 – 2003
- Various industrial roles in IT: Prior to 2001

Research in software architecture,
Service Orientation, Cloud Computing, and
Software Development Paradigm

http://malibabar.wordpress.com



IT UNIVERSITY OF COPENHAGEN

ITU, CPH

IT UNIVERSITY OF COPENHAGEN

Today's Talk

- What is Agility?
- Perceptions about architecture
- What is architecture?
- Why do we combine agile and architecture?
- Lessons from two case studies
- Some practical points on integration
- Take-Away – one thought
  - Agility and architecture:

    A match made in Heaven…broken on Earth?

Agility

- Agility is the ability to both create and respond to change in order to profit in a turbulent business environment.

Jim Highsmith (2002)

- Characteristics of Agile development
  - Iterative and incremental
  - Small releases
  - Release plan/feature backlog
  - Iteration plan/task backlog
  - Collocation

Sanjiv Augustine (2004)

Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions ***over process and tools***,
- Working software ***over comprehensive documents***,
- Customer collaboration ***over contract negotiation***,
- Responding to change ***over following a plan***.

That is, while there is value in the items on the right, **we value the items on the left *more***

Source: http://www.agilemanifesto.org/

Perceptions about Architecture

- Architecture is Big Up Front Design (BUFD)

- Architecture means massive documentations

- Architecture doesn’t add value to customers
  - ***You Ain't Gonna Need It*** (YANGI)

- Architect – Prescriptive guy

Agility

Architecture

What is Software Architecture?

- Architecture is the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution. (IEEE1471 – 2000).

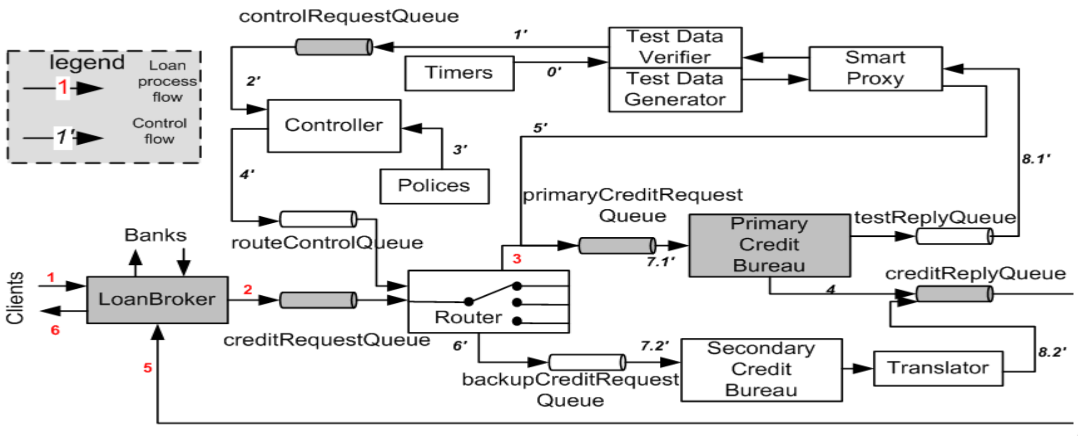
- A software system’s architecture is the set of principal design decision ma n (Taylor, R., et al., 2010)

- Its all a d and better c

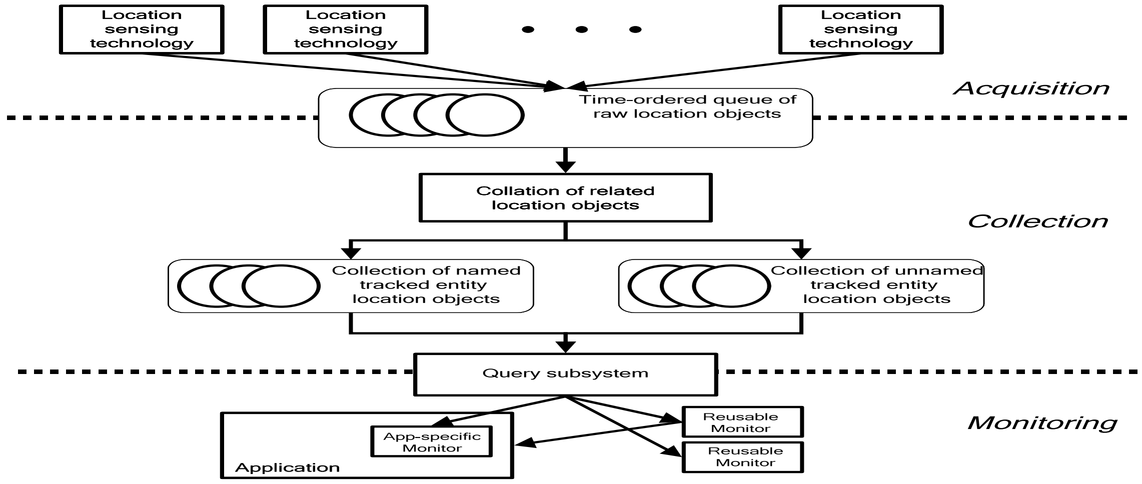
- Context – good decisions may become the bad ones

Software architecture should provide intellectual control and specifications for meaningful reasoning by stakeholders

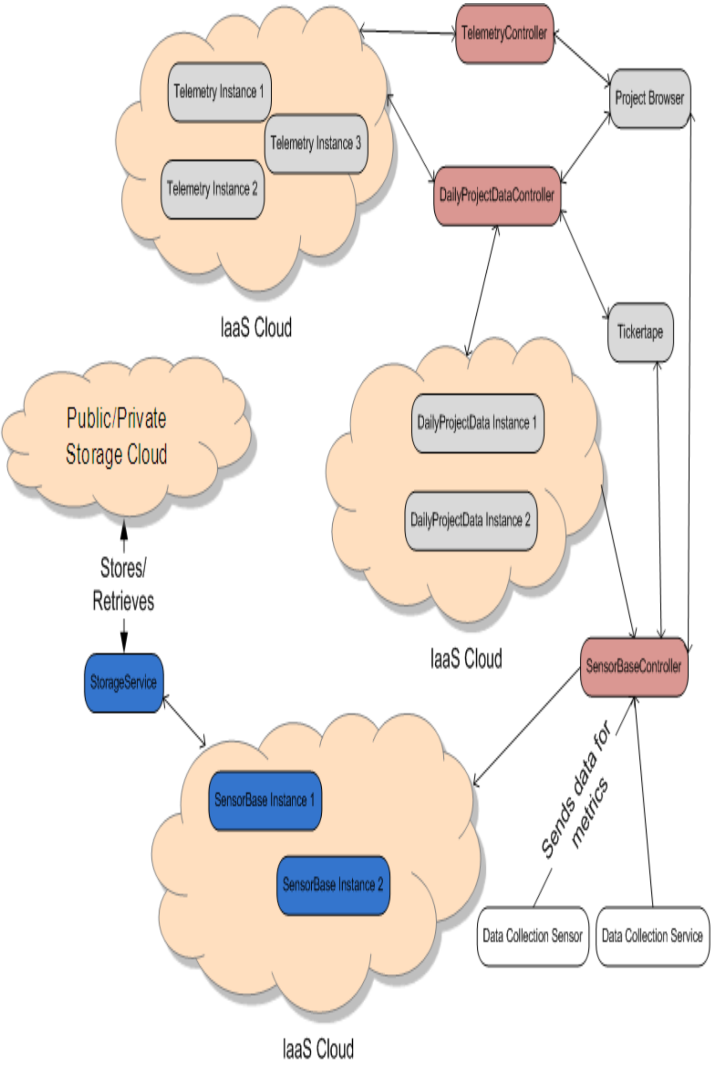
Architecture: Key Design Decisions



Source: Liu, Ali Babar, Gorton, QoSA 2008.


Location sensing technology
Location sensing technology
. . .
Location sensing technology
Time-ordered queue of raw location objects
Acquisition
Collation of related location objects
Collection
Collection of named tracked entity location objects
Collection of unnamed tracked entity location objects
Query subsystem
Reusable Monitor
App-specific Monitor
Application
Reusable Monitor
Monitoring

Source: Cooney et al., 2007



Quotes from Agile Practitioners!!!

- “*It seems that many agile method users misunderstand what agile methods are, just* ***ignore*** *architecture,* ***and jump onto refactorying****.*” *Satoshi Basaki*
- *“The* ***YAGNI*** *belief has led many agile team ultimately to a point of failure by ignoring the architecture’s essential elements.” Blair, Watt, Cull.*
- *“Architecture is just as* ***IMPORTANT*** *in XP projects as it is in any software project. Part of the architecture is captured by the system metaphore.” Kent Beck*
- “*Tension between agility and architecture might be* ***FALSE dichotomy****.” Craig Larman*

Augmenting XP: Why and How?

- Quality requirements

"***A system isn't certifiably secure unless it has been built with a set of security principles in mind and has been audited by a security expert. While compatible with XP these practices have to be incorporated into the team's daily work.***" (Kent Beck, 2004)

- Scaling XP

"***With awareness and appropriate adaptations, XP does scale. Some problems can be simplified to be easily handled by a small XP team. For others, XP must be augmented. The basic value and principles apply at all scales. The practices can be modified to suit your situation***."

- **Context based adaptation is INEVITABLE**

How to combine Agility & Architecture?

A Story….

- A market leader in financial products & services
- Multiple development sites with various development paradigms
- Agile adoption started in 2005
- Needed to combining plan driven and agile in distributed arrangements
- Main motivation was increased competition from other sites for internal offshoring

Architecture Design

- Agile project apply two stages of design solutions:
  - Draw HIGH LEVEL roadmap called Software Architecture Overall Plan (SAOP)
  - Developers look for flaws – design validation

- NO attention to quality attributes – rather use
  - Re-factoring – for example improving performance
  - Maintenance projects – can be up to 2 years!!!

- Upfront design – Something that would change later

- Main drivers - functionality, delivery time, budget

Architecture Documentation

- Before Agile
  - Comprehensive documentation of architecture and design
  - Minimum four weeks on specifications for a medium size project

- After Agile
  - Drastic reduction in architectural documentation – ONLY SAOP

- Argument against documentation - Formal documentation did not add much value to customers

- 30% - 40% reduction in documentation resources

- NO argumentation around and documentation of design that may NOT be implemented later on

Sharing Design Decisions

- Before Agile
  - Detailed architectural documentations and ARB meetings

- After Agile
  - Wiki and design meetings for sharing design decisions

- Design decisions on Whiteboards until implemented

- Wiki is delivered with software release

- Wiki based sharing of design initially works but then searching design decisions becomes cumbersome

- Tracking architectural decisions becomes hard

Agile Approaches – Positives

- Bringing developers EARLY in the design decisions

- Don't spend HUGE AMOUNT of time discussing and documenting solutions that may not be implemented

- Clear and agreed upon deliverables for KNOWN delivery date and budget - small iterations

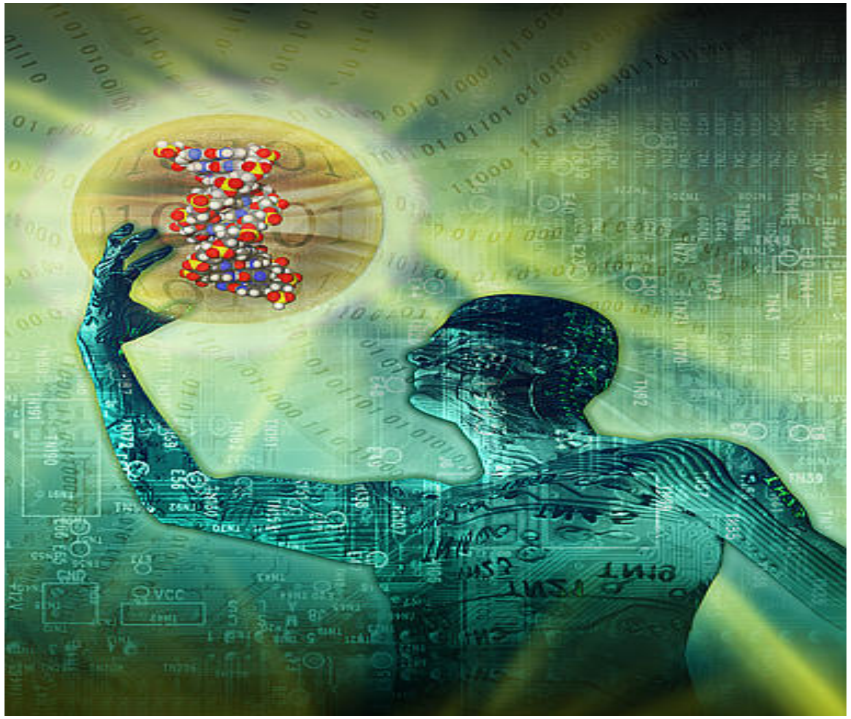
- Saving up to 30-40% resources on design documents

- EASILY and QUICKLY sharing design decisions and knowledge through Wikis and design meetings

Agile Approaches – Negatives

- Implementing User Stories WITHOUT a good knowledge of subsequent inter-dependencies
- Architecturally very RISKY for new projects when potential solutions are NOT very well understood
- NO time for careful design or considering alternatives
- NO encouragement to focus on quality attributes
- Design knowledge remains with INDIVIDUALS
- Searching design decisions on Wiki can be DIFFICULT

Challenges & Strategies!!!



Challenges and Strategies 1/2

- Incorrect prioritization of user stories (C)

- Involve architects and developers in feature analysis workshop (S)

- Lack of time and motivation for considering design choices (C)

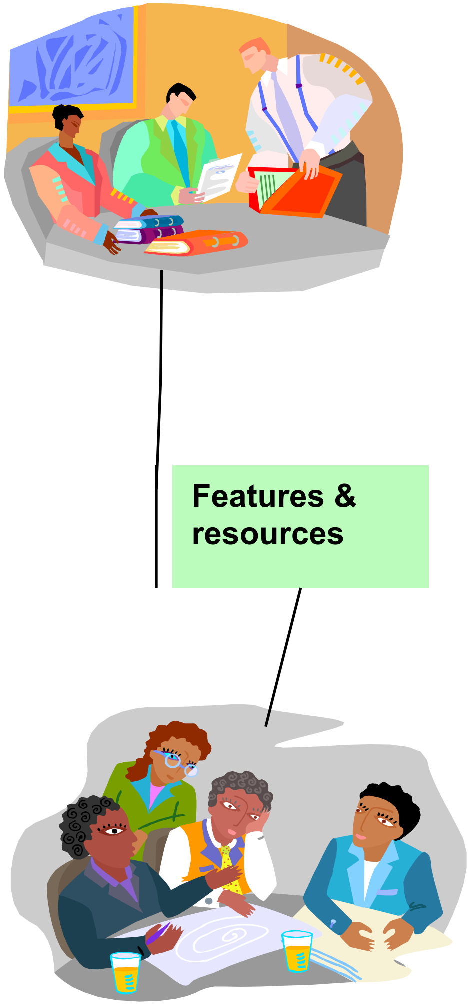
- Combine zero feature release with Feature Analysis Workshop (S)
  - Zero feature release - Do architecturally focused work without delivering any user-visible features

Challenges and Strategies 2/2

- Unknown domain and untried solutions (C)
- Apply hybrid approach (S)
- Pilot project for sorting out backlogs (S)
- Lack of focus on quality attributes (C)
- Make quality attributes a success factor (S)
- Link development and maintenance budgets (S)
- Lack of Skilled people (C)

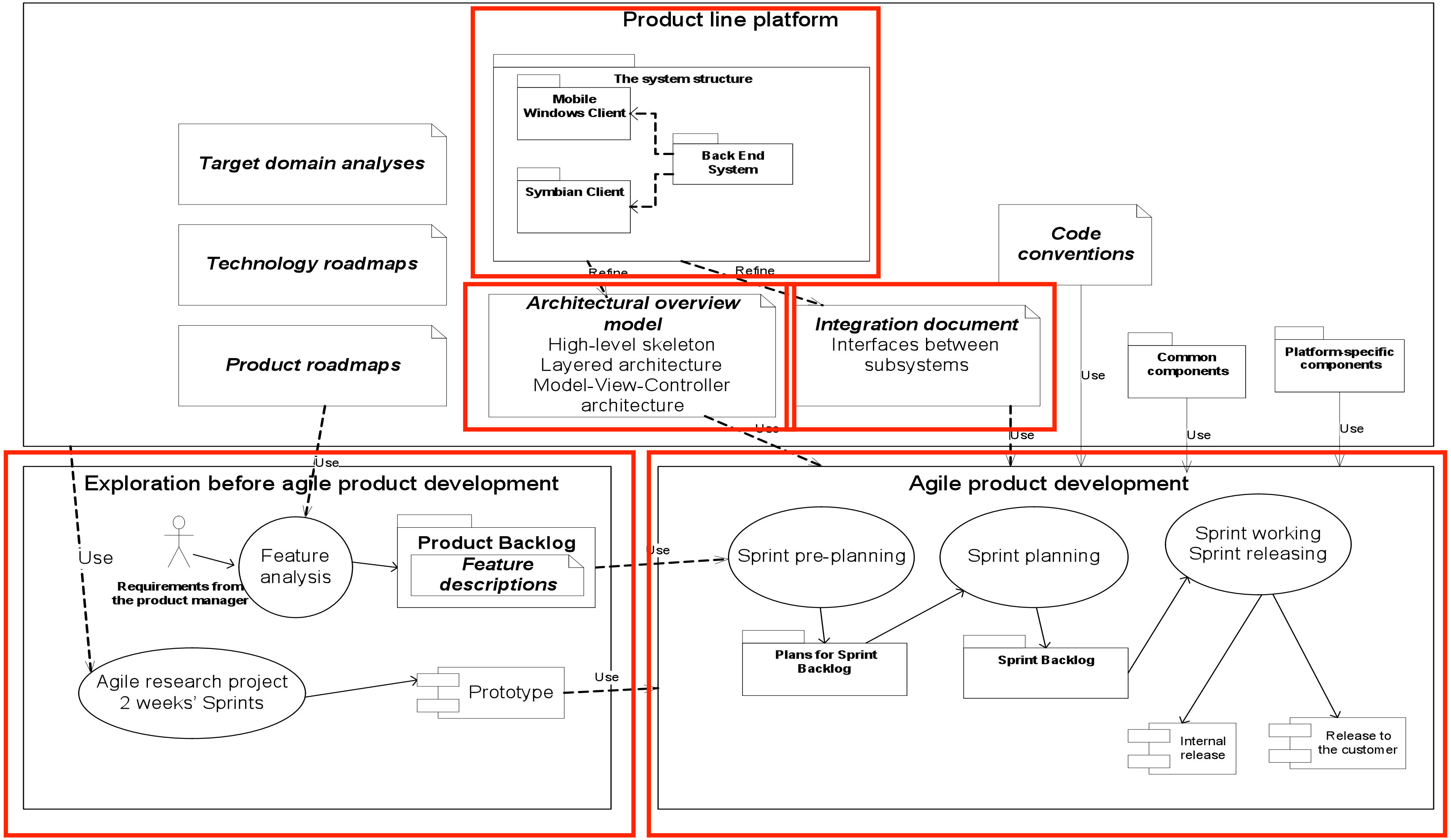
Another Story….

- Security software leader
- Market of 90+ countries
- Agile transformation begin in 2005
- Commonly held agile beliefs couldn’t work!!!
- Introduced platform based development for SPEED
- Agile & Product lines



Features & resources

Agile Approaches in Product Lines



Product line platform

The system structure

Mobile Windows Client

Back End System

Symbian Client

Target domain analyses

Technology roadmaps

Product roadmaps

Refine

Refine

Architectural overview model
High-level skeleton
Layered architecture
Model-View-Controller architecture

Integration document
Interfaces between subsystems

Code conventions

Common components

Platform-specific components

Use

Use

Use

Use

Use

Use

Exploration before agile product development

Use

Requirements from the product manager

Feature analysis

Product Backlog
Feature descriptions

Agile research project
2 weeks' Sprints

Prototype

Use

Use

Agile product development

Sprint pre-planning

Sprint planning

Sprint working
Sprint releasing

Plans for Sprint Backlog

Sprint Backlog

Internal release

Release to the customer

Key Practices 1/2

- Implementing features without up-front design exploration Doesn't work

- Research projects can discover potential problems

- Rotate staff between research and product projects

- Research projects are carried out using Agile practices BUT no delivered functionality
  - Shorter lengths of Sprints – 2 weeks

- Organize teams based on the use of platforms

Key Practices 2/2

- Establishing mutual trust between the lead architect and a project architect is essential
- Use of “Daily Meetings” for architectural discussions
- Use high level architectural description for subcontractors, new team members, big architectural modifications, and developing new products
- Each of the platforms has its own confluence to share architectural documents and knowledge

Communicating Architecture

- Communicating architectural knowledge is an integral part of integrating product line and Agile practices

- All designers regularly read the overall architecture and comments on debatable issues

- Every new designer is expected to read the whole lot from the beginning to the end and all updates

- Sharing architectural knowledge by locating all platforms’ teams very close to each other

A few more practical points

Architect: Role & Responsibilities

An architect should know how to sell a key design decision to product owners in conflicting situations

Institutionalized the role of architect with more focus on facilitation & serving

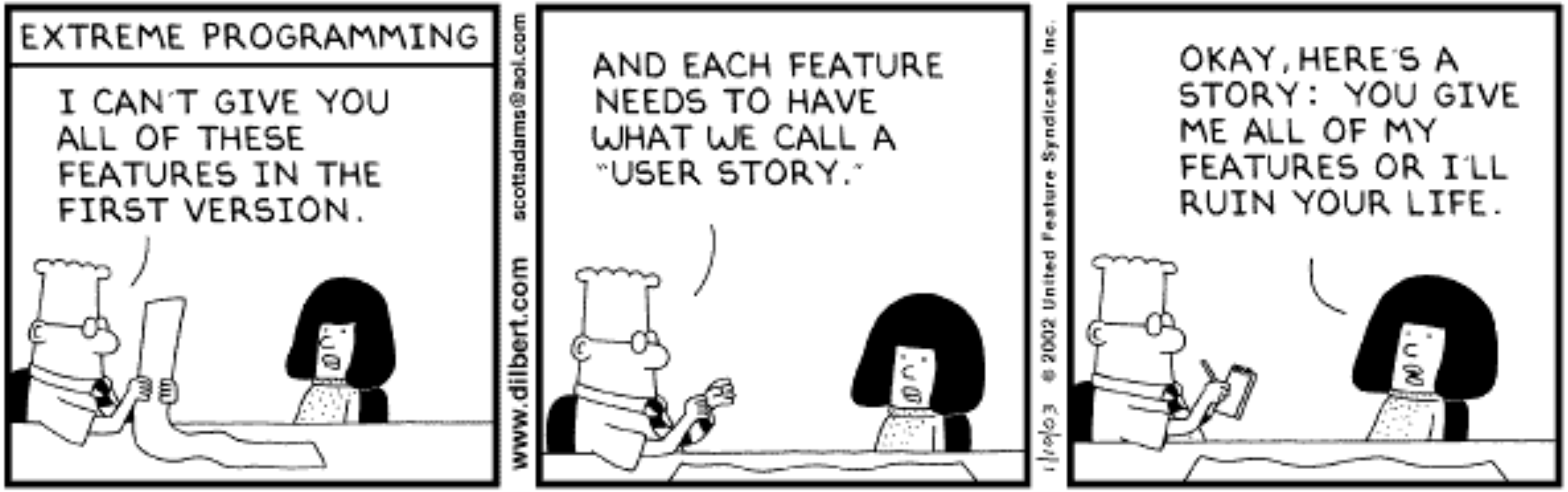
Project architect should know the overall architecture, required features, and implementation status

An architect needs to have good understanding of Agile approaches

Have multiple architects – solution architect, software architect and implementation architect for certain kinds of projects

Architect should document/update and communicate the architecture

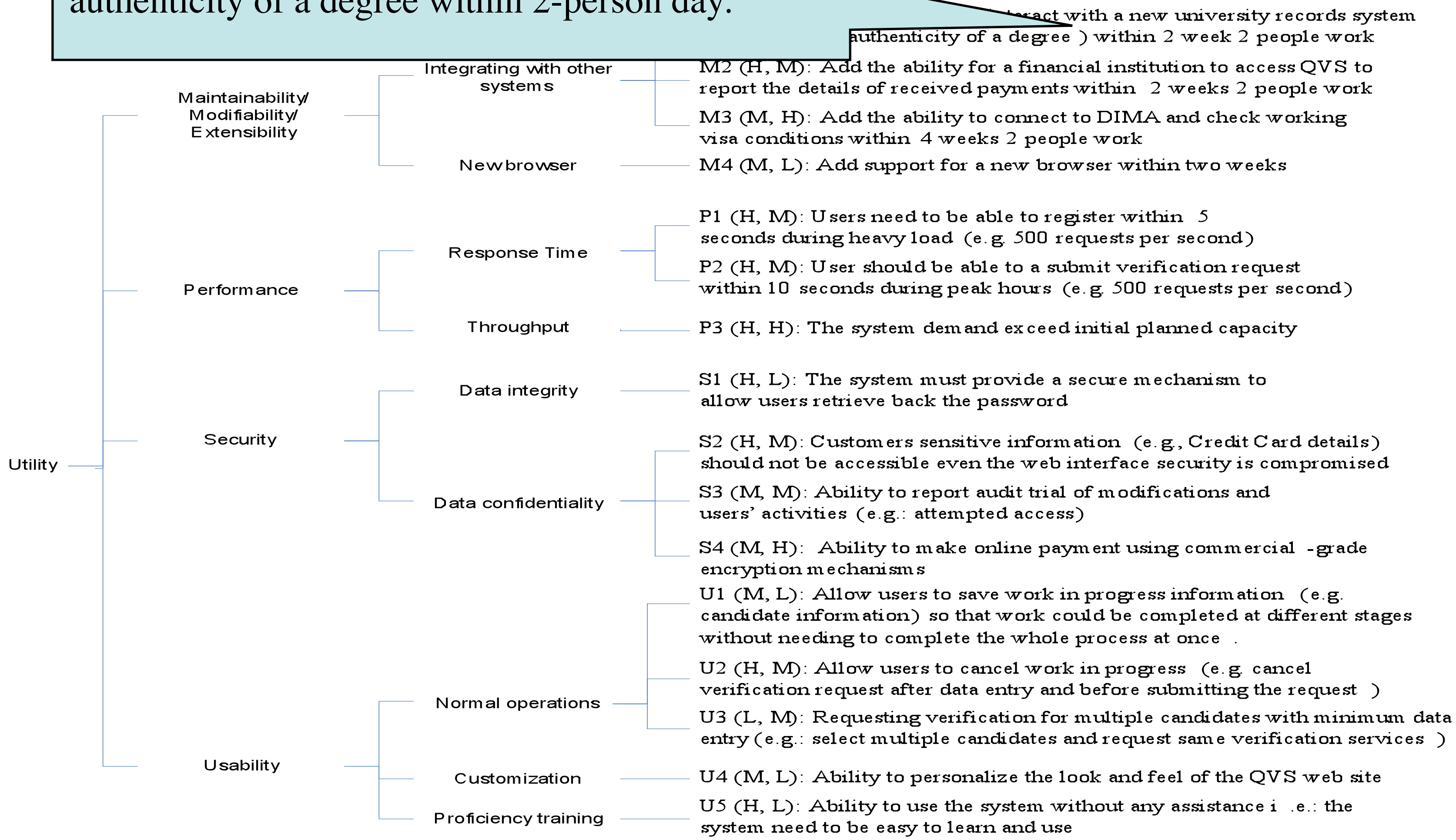
Users Stories….


EXTREME PROGRAMMING
I CAN'T GIVE YOU ALL OF THESE FEATURES IN THE FIRST VERSION.
scottadams@aol.com
www.dilbert.com
AND EACH FEATURE NEEDS TO HAVE WHAT WE CALL A "USER STORY."
1/10/03 © 2002 United Feature Syndicate, Inc.
OKAY, HERE'S A STORY: YOU GIVE ME ALL OF MY FEATURES OR I'LL RUIN YOUR LIFE.

Copyright © 2003 United Feature Syndicate, Inc.

User Stories + Quality Scenarios

M1 (H, H): Add the ability to interact with a new University record system to validate the authenticity of a degree within 2-person day.

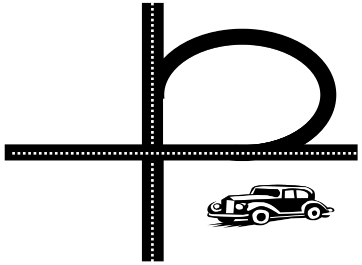


Utility

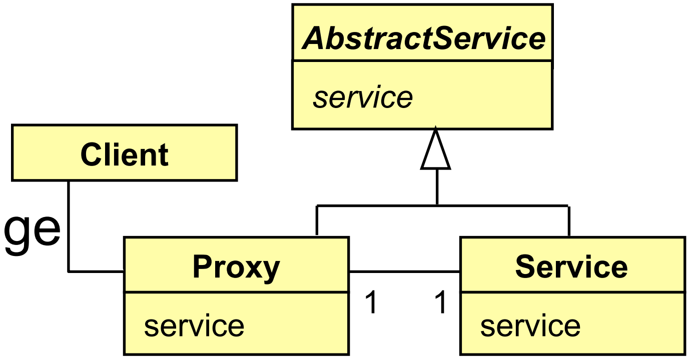
- Maintainability/ Modifiability/ Extensibility
  - Integrating with other systems
    - …eract with a new university records system … authenticity of a degree) within 2 week 2 people work
    - M2 (H, M): Add the ability for a financial institution to access QVS to report the details of received payments within 2 weeks 2 people work
    - M3 (M, H): Add the ability to connect to DIMA and check working visa conditions within 4 weeks 2 people work
  - New browser
    - M4 (M, L): Add support for a new browser within two weeks
- Performance
  - Response Time
    - P1 (H, M): Users need to be able to register within 5 seconds during heavy load (e.g. 500 requests per second)
    - P2 (H, M): User should be able to a submit verification request within 10 seconds during peak hours (e.g. 500 requests per second)
  - Throughput
    - P3 (H, H): The system demand exceed initial planned capacity
- Security
  - Data integrity
    - S1 (H, L): The system must provide a secure mechanism to allow users retrieve back the password
  - Data confidentiality
    - S2 (H, M): Customers sensitive information (e.g., Credit Card details) should not be accessible even the web interface security is compromised
    - S3 (M, M): Ability to report audit trial of modifications and users' activities (e.g.: attempted access)
    - S4 (M, H): Ability to make online payment using commercial -grade encryption mechanisms
- Usability
  - Normal operations
    - U1 (M, L): Allow users to save work in progress information (e.g. candidate information) so that work could be completed at different stages without needing to complete the whole process at once .
    - U2 (H, M): Allow users to cancel work in progress (e.g. cancel verification request after data entry and before submitting the request)
    - U3 (L, M): Requesting verification for multiple candidates with minimum data entry (e.g.: select multiple candidates and request same verification services)
  - Customization
    - U4 (M, L): Ability to personalize the look and feel of the QVS web site
  - Proficiency training
    - U5 (H, L): Ability to use the system without any assistance i .e.: the system need to be easy to learn and use

Exploit Scenarios & Patterns

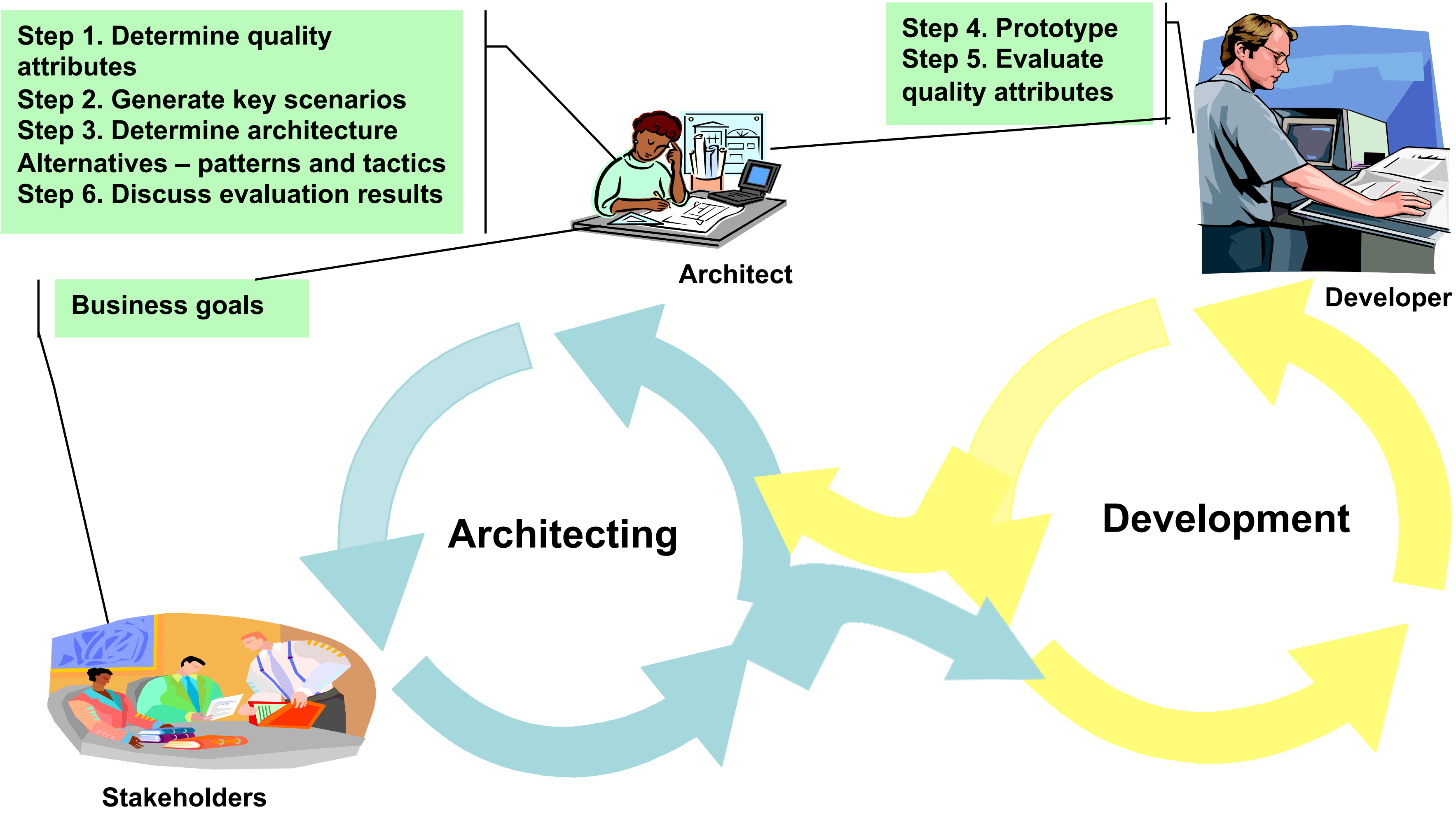
- Scenarios are useful for evaluating multiple quality attributes of software architecture
- Key scenarios can drive the evaluation
  - describe the behavior of architecture
  - set the context for particular quality attributes



- Knowledge of patterns is always handy for quickly evaluating design alternatives
- lightweight and agile process
  - Only two roles involved
  - Repository of architectural knowledge



Agile Evaluation of Architecture

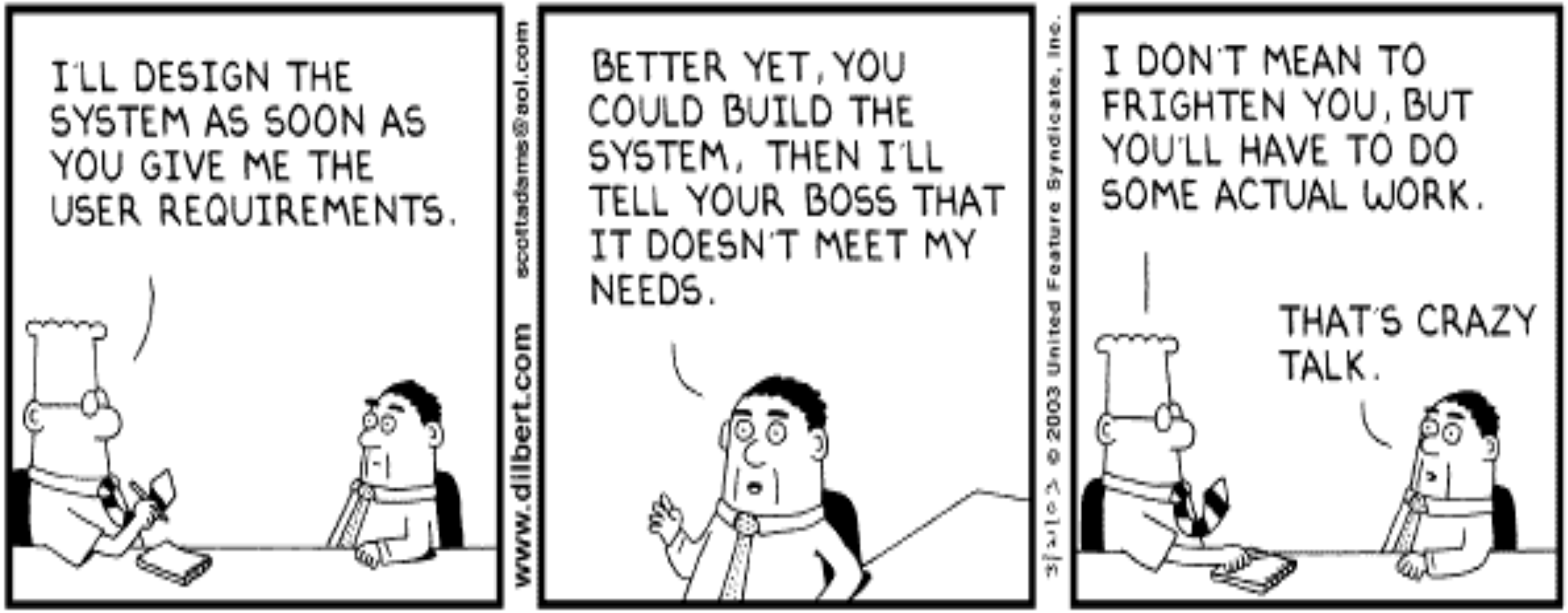


Step 1. Determine quality attributes
Step 2. Generate key scenarios
Step 3. Determine architecture Alternatives – patterns and tactics
Step 6. Discuss evaluation results

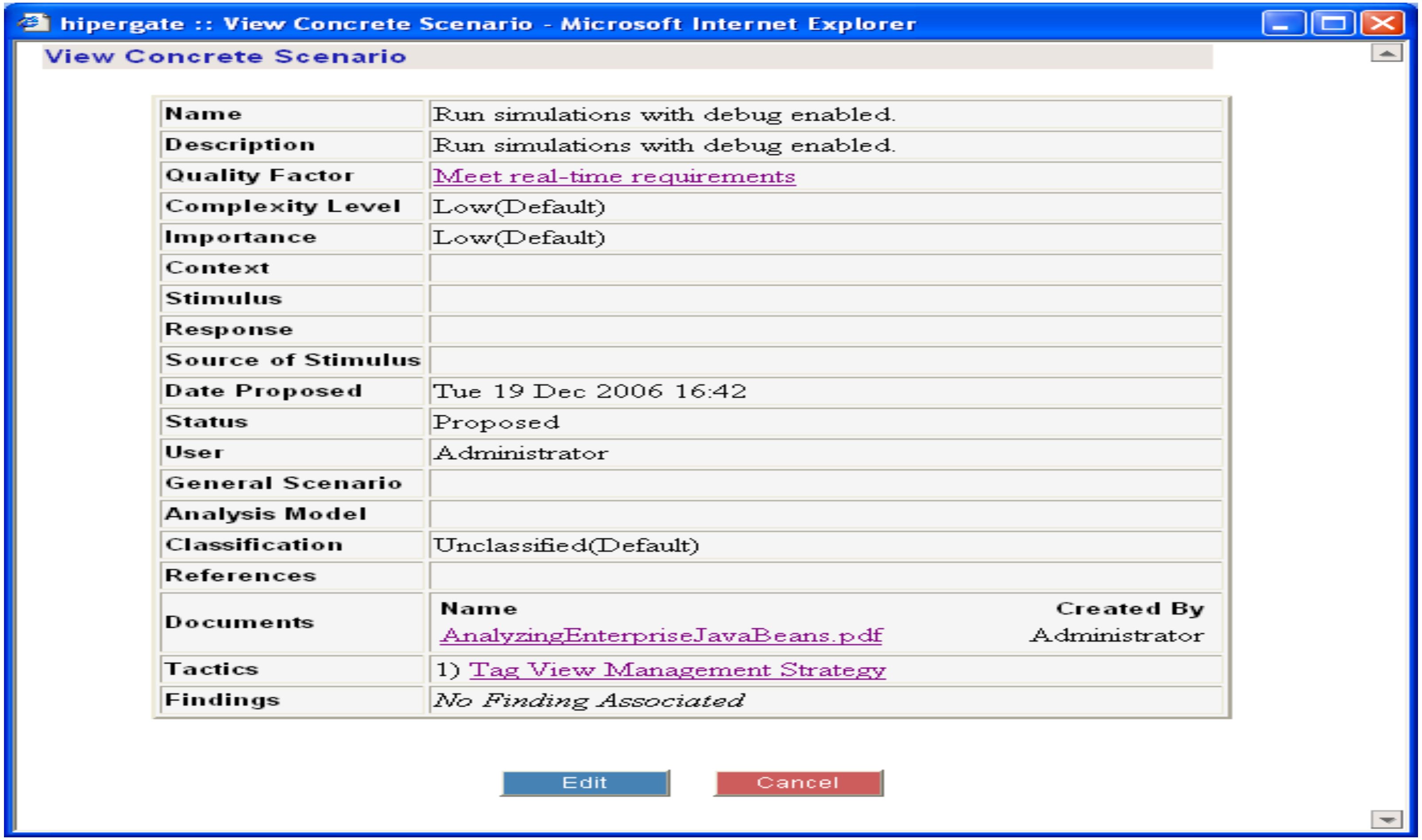
Step 4. Prototype
Step 5. Evaluate quality attributes

Business goals

Get Stakeholders on Board Early


I'LL DESIGN THE SYSTEM AS SOON AS YOU GIVE ME THE USER REQUIREMENTS.
scottadams@aol.com
www.dilbert.com
BETTER YET, YOU COULD BUILD THE SYSTEM, THEN I'LL TELL YOUR BOSS THAT IT DOESN'T MEET MY NEEDS.
© 2003 United Feature Syndicate, Inc.
I DON'T MEAN TO FRIGHTEN YOU, BUT YOU'LL HAVE TO DO SOME ACTUAL WORK.
THAT'S CRAZY TALK.

Design and Use Simple Templates



hipergate :: View Concrete Scenario - Microsoft Internet Explorer

View Concrete Scenario

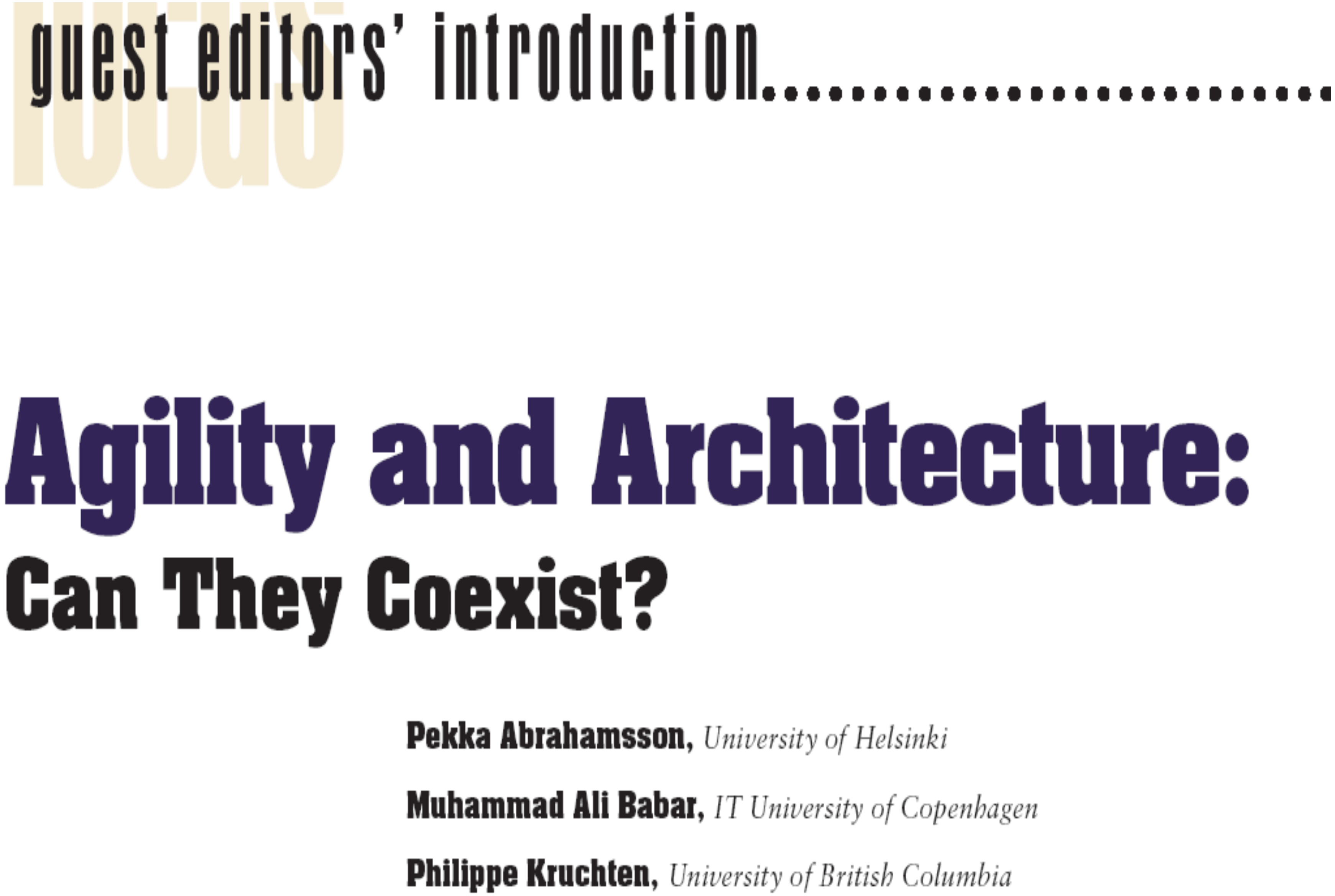
Name	Run simulations with debug enabled.
Description	Run simulations with debug enabled.
Quality Factor	Meet real-time requirements
Complexity Level	Low(Default)
Importance	Low(Default)
Context	
Stimulus	
Response	
Source of Stimulus	
Date Proposed	Tue 19 Dec 2006 16:42
Status	Proposed
User	Administrator
General Scenario	
Analysis Model	
Classification	Unclassified(Default)
References	
Documents	Name: AnalyzingEnterpriseJavaBeans.pdf Created By: Administrator
Tactics	1) Tag View Management Strategy
Findings	*No Finding Associated*

Agile Values and Architecture

XP values	Architectural Approaches
Communication	Facilitate stakeholders' involvement at all stages of development
Simplicity	Coarse-grained design with only enough architecting to ensure quality attributes
Feedback	Architectural evaluation provides early feedback on risky and non-risky decisions
Courage	Foreseen changes can be planned and incorporated in the design, risk avoidance

A Few Take-Aways!!!

- Understand the Context
- Clearly and precisely define architecture
- Show architecture’s business value to product owner
- Communicate and coordinate through architecture
- Use critical functionality to assess architecture
- Understand when to freeze the architecture
- Track unresolved architecture issue (backlog)



focus

guest editors' introduction

Agility and Architecture: Can They Coexist?

Pekka Abrahamsson, *University of Helsinki*

Muhammad Ali Babar, *IT University of Copenhagen*

Philippe Kruchten, *University of British Columbia*

Acknowledgements

- Discussions with Philippe Kruchten and his writings and ideas shared by Pekka Abrahamsson
- Collaboration with Minna Pikkarainen and Toumas Ihme of VTT, Finland was the main sources of case studies
- Some ideas have been based on the articles submitted to our call to a special issue of IEEE Software and included in its final publication in March/April, 2010.

References

- Abrahamsson, P., Ali Babar, M., Kruchten, P., Agility and Architecture: Can They Coexist?. IEEE Software 27(2): 16-22 (2010).
- Faber, R., Architects as Service Providers. IEEE Software 27(2): 33-40, (2010).
- Madison, J., Agile Architecture Interactions. IEEE Software 27(2): 41-48, (2010).
- Blair, S., Watt, R., Cull, T., Responsibility-Driven Architecture. IEEE Software 27(2): 26-32, (2010).
- Ali Babar, M., An exploratory study of architectural practices and challenges in using agile software development approaches. WICSA/ECSA 2009: 81-90.
- Ali Babar, M., Ihme, T., Pikkarainen, M., An industrial case of exploiting product line architectures in agile software development. SPLC 2009: 171-179.
- Nord, R., Tomayko, J., Software Architecture-Centric Methods and Agile Development. IEEE Software 23 (2): 47-53 (2006).
- Hofmeister, C., Kruchten, P., Nord, R., Obbink, H., Ran, A., America, P., A general model of software architecture design derived from five industrial approaches. Journal of Systems and Software 80(1): 106-126 (2007).

Thank You

M. Ali Babar

alibabar.m@gmail.com
malibabar.wordpress.com