THE IRISH SOFTWARE
ENGINEERING RESEARCH CENTRE

Architectural Practices and Challenges in Using Agile Software Development Approaches

M. Ali Babar

Today's Talk

- Agility and architecture:

A match made in Heaven…broken on Earth?

- Talk summarizes
  - The design, logistics, and results from a couple of studies aimed at understanding architectural practices of Agile teams
  - Our own experiences, observations, and thoughts

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
  - Iterative and incremental
  - Small release
  - Collocation
  - Release plan/ feature backlog
  - Iteration plan/task backlog

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/

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. For example, re-factorings have to preserve the security of the system as well as its functionality***" (Kent Beck, 2004)

- Scale

"***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 is paramount

Software Architecture: Definitions

Software Architecture (SA) is the structure or structures of the system, which comprise software components, the externally visible properties of these components and relationships among them. (Bass et al., 2003)

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 architecture should be defined in terms of elements that are coarse enough for human intellectual control and specific enough for meaningful reasoning.

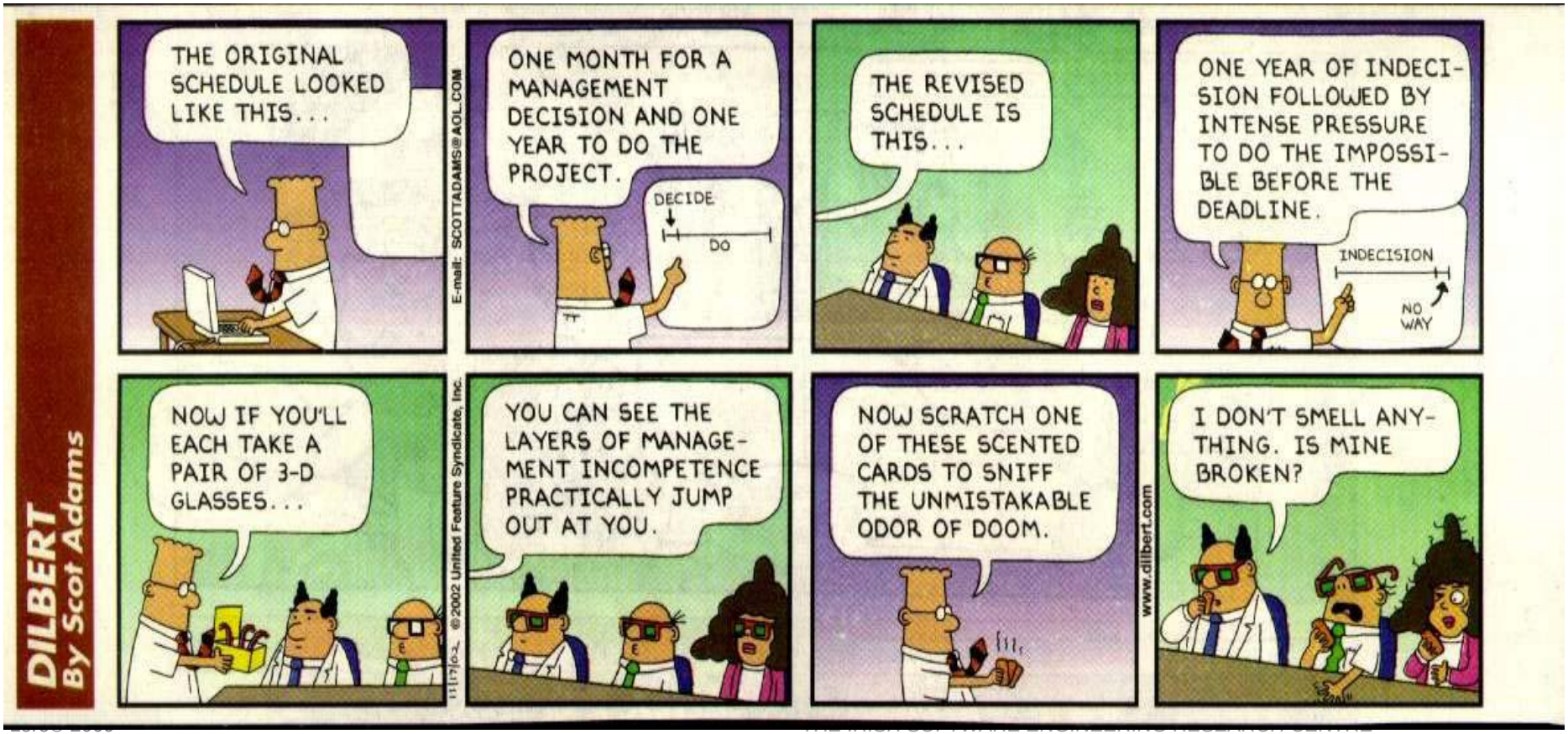
(Kazman et al., 2006)

- Its all about DECISIONS – bad, good and better ones
- Context – good decisions may become the bad ones

Why is Architecture Hard?

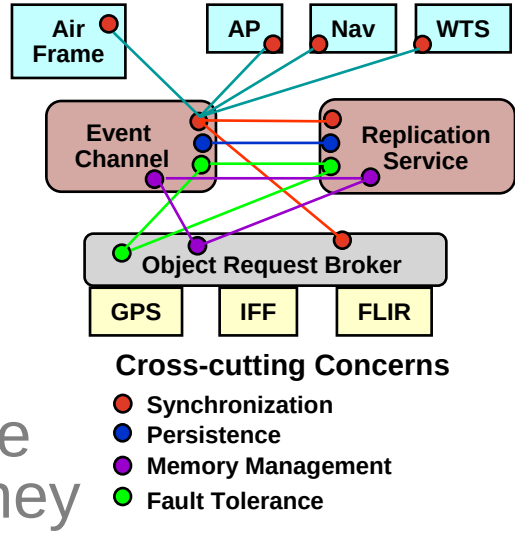
"..The life of a software architect is a long (and sometimes painful) succession of sub-optimal decisions made partly in the dark…"

(Philippe Kruchten)



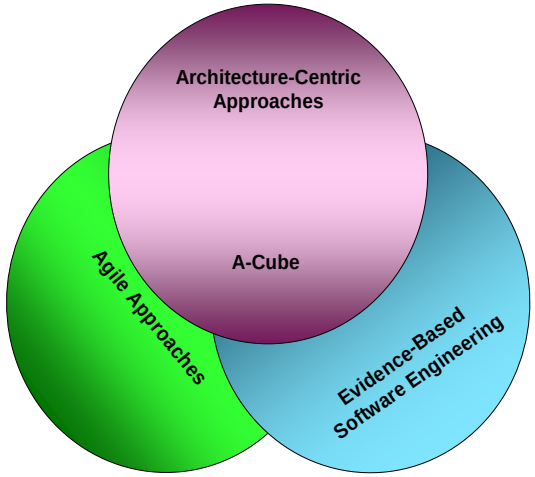
Architectural Design Decisions

- Architecture design typically takes place at an early stage
  - hard, often impossible, to thoroughly reason about the consequences of many design decisions
- Involves making design decisions that are difficult/costly to change downstream if they are discovered to be flawed
- Complex design trade-offs need to meet competing architectural requirements
- Put very simply – architecture aims to address any issues that will be expensive/impossible to change once the project progresses



Background of This Research

- Research motivation
  - Lack of sufficient attention to architectural issues in Agile approaches makes their scalability questionable
  - Bridging the gap between Architecture and Agile is essential
- Research Goals
  - Develop and/or customize approaches and tools to help companies to integrate sound architectural principles and agile approaches
  - Exploit best practices in the areas of software architecture, agile methods, and evidence-based software engineering

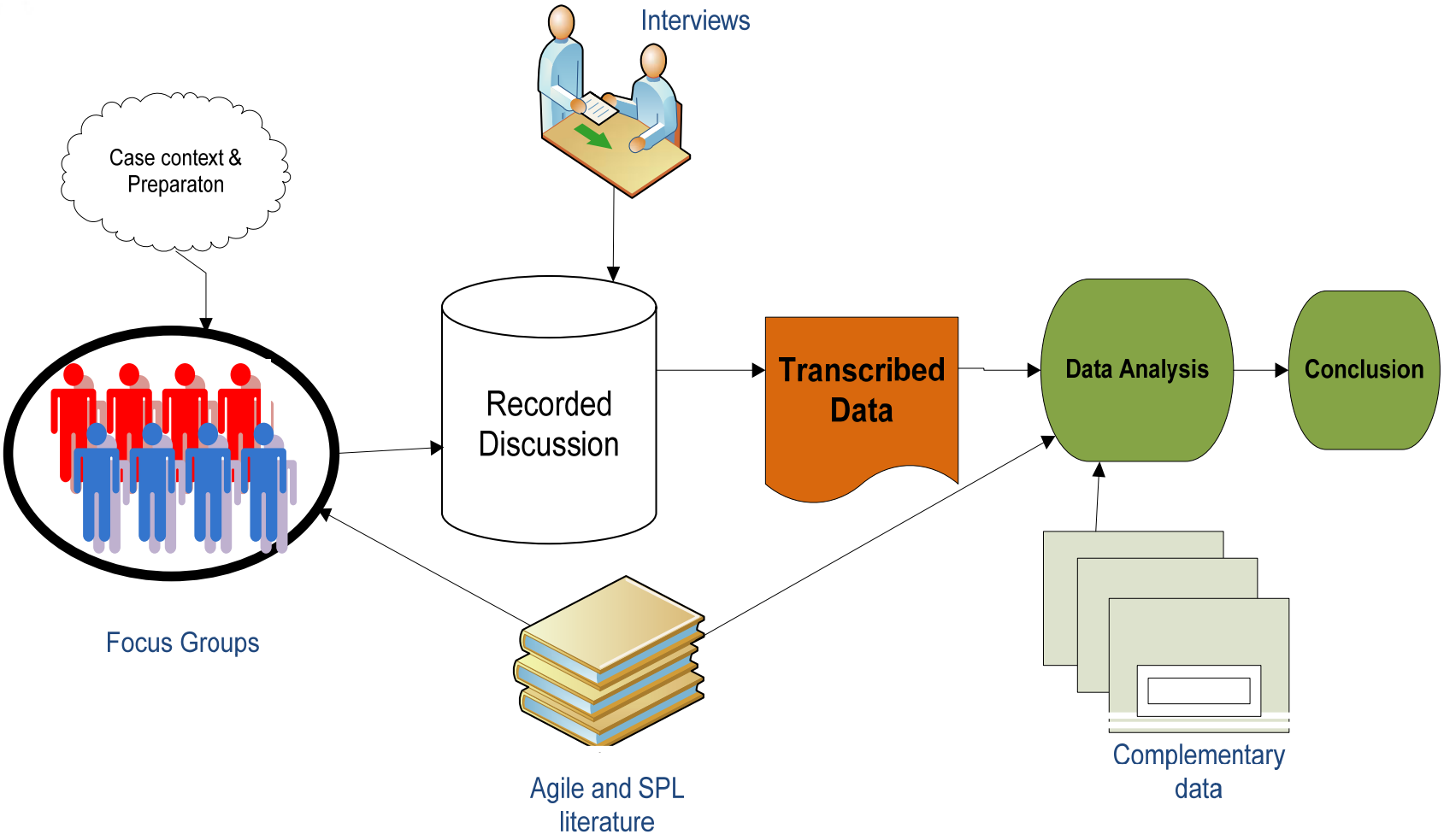


http://www.acube-ommunity.org

This Research….

- Main objectives are:
  - To understand the role and importance of software architecture within teams using agile approaches
  - To determine the architecture-related practices and challenges of agile teams and potential solutions to deal with those challenges
- Contributions
  - Design and logistics of an empirical study
  - Empirically founded information about how agile teams approach architectural aspects
  - Challenges and potential solutions to deal with problems caused by architecture-related issues in agile approaches

An Overview of Research Process



Research Design

- Research Methodology
  - Case Study
- Data Collection Methods
  - Focus group sessions
  - Interviews
  - Observations from the complementary data
- Data analysis
  - Transcription
  - Content analysis

Case Study Methodology

- Pre-planning or research initiation
- Administration
- Planning or focus the case study
- Design case study plan
- Data collection
- Data analysis
- Reporting

Study Procedure

- Intended to organize separate focus groups of:
  - Technical staff (i.e. software architects)
  - Management staff (i.e. project or team manager)
- Participants of two focus groups: four software architects and three managers
- Interviews with technical leads of three projects
- Timeline and duration
  - Focus groups and interviews were conducted in September 2008 – Focus groups lasted 90 and 60 minutes and interviews lasted around 30 minutes.

Demographics

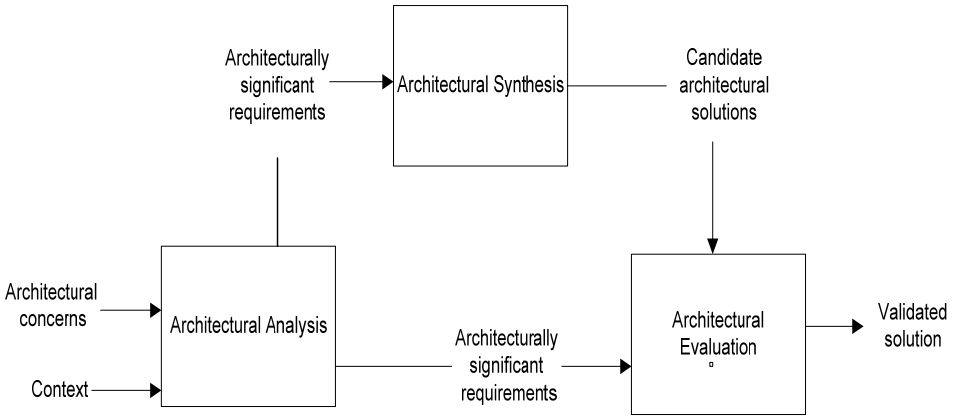
- Company traditionally used waterfall, iterative, plan-driven development processes
- Study site adopted agile in 2006 and became part of a large European project on Agile processes
- Focus group participants
  - On average 8 years of working experience
  - Shortest project 4 months and longest 4 years
- Interviewees
  - Average work experience of 5 years
- All participants had worked with both agile and plan driven approaches
- All participants had worked in distributed as well as co-located arrangements

Data Analysis Framework

- Architect's role and responsibilities
- General Model of Software Architecture Design
- Architecture documentation
- Communicating design decisions
- Challenges and strategies

A Model of Architecture Design

- Three main stages
  - Architecture Analysis
  - Architectural Synthesis
  - Architectural Evaluation
- Several inputs and outputs
  - Architectural concerns
  - Context
  - Architecturally significant requirements
  - Candidate solutions
  - Validated architectural solutions
- Has concept of backlogs



Source: Hofmeister et al., 2006.

Architect's Role & Responsibilities

Offsite architect designs Software Architectural Overall Plan (SAOP) that provides the technical roadmap

Role of architect was institutionalized before the adoption of Agile approaches & it got modified

Architect documents/updates and communicates the architectures

Architect resides offshore with clients to acquire and prioritized User Stories

Solution architect takes up more management-oriented role including Scrum Master

Implementation architect is responsible for getting User Stories implemented, mentoring developers, re-factoring

Architectural Analysis

- Most of the tasks related to analysis (e.g., examining context and defining problems) had been pushed towards offshore clients
- Drawing a high level architectural roadmap and writing User stories
- Inspecting User stories for detailed design decisions
- Design decisions are based on delivering features within fixed cost and time
- NO attention to quality attributes as they are NOT the measure of success
- Maintenance projects FIX the quality attributes

Architecture Synthesis

- Previously used proprietary design methodology with all kinds of deliverables and artefacts
- Agile project apply two stages of design solutions:
  - Software architects work with clients to draw HIGH LEVEL roadmap called Software Architecture Overall Plan (SAOP)
  - Solution and implementation architects would make design decisions for implementing User Stories
  - Limited number of design options are considered
- Compared to General Design Model, Agile teams produce significantly LESS number of artefacts
- All the deliverables are made available on Wiki that is handed over to the maintenance projects and clients
- The main deliverable is SAOP

Architectural Evaluation

- Architecture evaluation used to be a formal process with Architecture Review Board (ARB)
- NOW, developers are asked to look for flaws in the architecture – more like design validation rather than architecture evaluation
- Re-factoring can FIX quality attributes
  - Architects agree that large-scale re-factoring is risky
  - Developers believe re-factoring is much better in fixing quality attributes than upfront design
- Argument against upfront design – what is the point of doing something that has to be changed at the implementation?
- Main drivers are functionality, delivery time, budget

Architecture Documentation

- "*Working software over comprehensive 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
  - Wiki for sharing design decisions
- Argument against documentation - Formal documentation did not add much value to customers
- 30% - 40% reduction in resources required for documentation
- Getting into development has advantages - NO argumentation around design that may not be implemented later on

Communicating Design Decisions

- Before Agile
  - Architectural and design documentations
  - Architecture Review Board meetings
- After Agile
  - Wiki and design meetings for sharing design decisions
  - Wikis without any formal templates and structure
- Design decisions captured on Whiteboards until implemented
- Customers and maintenance project teams get access to Wiki when software is released
- Wiki based sharing of design decisions appears to work well initially but then searching design decisions becomes cumbersome
- After sometime, tracking architectural decisions becomes hard for impact analysis

Artefacts Used by Agile Teams

Activities	Artifacts	Artifacts used by the agile teams
Architectural analysis	Context	Platforms, fixed cost, fixed duration
	Requirements, Architecturally significant requirements	User stories focusing on features to be delivered No particular focus on quality attributes
Architectural synthesis	Candidate architectural solutions	Limited number of solutions known by the architecture team
	Architectural design (views, perspectives, prototypes).	Architectural Infrastructure plan (AIP) and TST documents mandated by company policy, User Stories
	Rationale	Rationale
Architectural evaluation	Quality attributes	Focus on features described by User Stories No particular focus on quality attributes
	Architectural assessment	Inter team cooperation for design assessment
Overall process driver	Backlog	Product backlogs and Sprint backlogs

Challenges and Strategies 1/2

- Incorrect prioritization of user stories (C)
  - Technical consideration not taken into account while prioritizing user stories
  - Critical interdependency among user stories requires large scale re-factoring
- Involving architects and developers in feature analysis works (S)
- Lack of time and motivation for considering design choices (C)
  - Achieve required features within time and budget
  - No upfront design, no consideration of alternatives
- Have Zero iteration among Agile followers (S)
- Combine Zero iteration with Feature Analysis Workshop (S)

Challenges and Strategies 2/2

- Unknown domain and untried solutions (C)
  - Agile may not be suitable when working in new domain, with new client, or with untried solution
  - Difficult start delivering features from first iteration
- Apply hybrid approach (S)
- Pilot project for sorting out backlogs (S)
- Lack of focus on quality attributes (C)
  - Architectural structure makes it hard to meet quality attributes
  - Achievement of quality attribute not a measure of success
- Achieving quality attributes a measure of success (S)
- Link budgets for development and maintenance projects (S)
- Lack of Skilled people (C)

Advantages of Using Agile Approaches

- Bringing developers EARLY in the picture for project design decisions
- NO need for spending HUGE AMOUNT of time on 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 architectural and design documentation activities
- EASILY and QUICKLY sharing design decisions and knowledge through Wikis and design meetings

Disadvantages of Using Agile Approaches

- Implementing User Stories WITHOUT a good knowledge of subsequent inter-dependencies of design decisions
- Architecturally very RISKY for new projects when potential solutions are NOT very well understood
- NO time for careful design during iterations
- NO considerations for alterative, potentially better design choices can be missed
- NO focus on quality attributes except some implicit focus on performance issues
- Design knowledge remains with the INDIVIDUALS
- Searching design decisions on Wiki can be DIFFICULT

Another Study….

- Focuses on the role of product line architectures in agile development teams
- Main objective
  - Empirically studying organizational processes and practices aimed at integrating SPL and Agile practices
- Research Methodology
  - Focus group
- Contributions
  - Provides information about and insights into the processes and practices of a company who has been leveraging product line architectures for improving its Agile software development

Study Procedure

- Intended to organize separate focus groups of:
  - Technical staff (i.e. software architects)
  - Management staff (i.e. project or team manager)
- Participants of two focus groups - three software architects and five managers
- Preliminary results were presented to the three platform and several product teams for comments
- Focus group sessions were carried out in English and feedback workshop was held in Finnish
- Timeline and duration
  - Focus groups in October 2008 and feedback workshop in early February 2009 – each lasted 2 hours

Demographics

- Company adopted agile processes in 2005
- Introduced project line platform strategy in 2007
- Software Architects
  - Had worked on 27 projects with architecture design experience ranging from 2 to 8 years
- Project managers
  - Average work experience of 15 years in technical and managerial roles
  - Had worked on dozens of projects
  - Scrum masters since 2005
- All participants had worked with both agile and plan driven approaches
- Feedback workshop was open to all members of the platforms and products development teams

Some of the Practices

- Architectural changes are identified in retrospectives
- Feature description documents are used to describe new features and modules and their architectural effects
- Feature description documents provide product projects with pre-planned iteration cycle, features and workload estimates
- Standardized coding conventions define micro architectures and guide the implementation of the architectures and interfaces
- Architectural documentation practices changed twice
  - No written architectural documents – proved quite risky
  - Writing short description of architectural decisions when needed, feature description documents and descriptions of the interfaces

Architectural Communication 1/2

- Communicating architectural knowledge is an integral part of integrating PL 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
- Avoiding distributed development

Architectural Communication 2/2

- A good working relationships and mutual trust between the lead architect and a project architect are the essential ingredient for integrating PL and Agile
- Use of “Daily Meetings” for architectural discussions
- Overall architecture description is very useful 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

Architectural Responsibilities

The lead architects work for research activities but do not participate in daily development activities of projects

Role of architect was institutionalized before the adoption of Agile approaches & remained unaffected

Architects need to have good understanding of Agile approaches

Project architect knows the overall architecture, required features, and implementation status

Project architect document/update and communicate the architectures

Key Points 1/3

- Objective was to study organizational processes & practices for integrating SPL and agile approaches
- Srum and XP assumptions about implementing features in short fixed-time iterations without any up-front design exploration may not be correct
- Agile development need documented architectures of platforms and sharing of tacit knowledge
- Use of research project for exploring potential problems between SPL and an intended Agile product development project

Key Points 2/3

- Research projects can also be used to study the feasibility of existing features during a SPL's maintenance or evolution
- Research projects are carried out using Agile concepts and practices (e.g., Scrum backlogs, Sprints), however, these projects may not qualify to be agile as there is no delivered executable business functionality
- Research projects have two weeks Sprints but product projects have four weeks Sprints

Key Points 3/3

- Introduce new roles for communicating architectural decisions and information: lead architect and team architect
- Minimize the need for cross-team communication by organizing teams according to the used software platforms in a product line
- Avoid distributed development
- Rotate the people between research and product development projects

Limitations

- Exploratory case studies to identify research questions and propositions
- Findings can only be generalizable to the population with similar characteristics to the participants
- Findings are mainly based on self-reported data – what people say, not necessarily what they actually observe, believe or perceive
- Small sample size, company, and country specific factors are other limitations of this study
- Broader representations of the practitioners and companies is required for future studies

Conclusions and Future Work

- Integration of development paradigms for synergies is inevitable – SPL, Agile, OSS, GSD….
- Understanding the impact of introducing agile on architectural processes is necessary for integrating architecture-centric methods and agile approaches
- These studies provide evidence that:
  - Adoption of agile approaches causes several changes in architectural processes with potentially negative effects
  - Some agile practices may have significant influence on architectural practices and artefacts
- Findings contribute to the growing body of knowledge about integrating Agile and Architectural approaches
- Looking for more companies interested in exploring the integration pros and cons

THE IRISH SOFTWARE
ENGINEERING RESEARCH CENTRE

Thank You