Building a Platform for Innovation: Architecture and Agile as Key Enablers

Peter Eeles
FSS Industry Lead, IBM Rational Worldwide Tiger Team
peter.eeles@uk.ibm.com
1191A
Please note the following

IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM’s sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.
Agenda

1. Introduction
2. Architecture
3. Agility
4. Building the Platform
5. Summary
Innovation

- Innovation: The action of innovating; the introduction of novelties; the alteration of what is established by the introduction of new elements or forms. [Oxford English Dictionary]
Top innovation goals

Superior product performance and product quality were seen as the most important goals by a plurality of innovators, with much less priority for other goals, such as the success rate of new products.

**Perceived Importance of Common Innovation Goals/Outcomes**

- Superior product performance
- Superior product quality
- Products customized to local markets and geographies
- Advantaged products and services
- Developing low-cost products
- Products developed for multiple markets
- Speed-to-market of product development and introduction
- Number of breakthrough products
- Success rate of new-product introductions

**Mean Ranking of Importance**

Source: Booz & Company
Key change enablers

The “What”
- Architecture
  - enables
  - enables
Innovation
- Change
  - enables
  - enables
The “How”
- Agility
The enterprise informs IT …
… and IT enables the enterprise

Enterprise

Architecture

enables

Change

enables

Agility

IT

Architecture

enables

Change

enables

Agility
## Similar concepts, differing scope

<table>
<thead>
<tr>
<th></th>
<th>Enterprise</th>
<th>IT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architecture</strong></td>
<td>Business operations</td>
<td>IT operations</td>
</tr>
<tr>
<td></td>
<td>Systems</td>
<td>Subsystems</td>
</tr>
<tr>
<td></td>
<td>Applications</td>
<td>Components</td>
</tr>
<tr>
<td><strong>Agility</strong></td>
<td>Business agility</td>
<td>Project agility</td>
</tr>
<tr>
<td></td>
<td>Transition planning</td>
<td>Project planning</td>
</tr>
<tr>
<td></td>
<td>Portfolio management</td>
<td>Backlog management</td>
</tr>
<tr>
<td></td>
<td>Strategy execution</td>
<td>Project execution</td>
</tr>
<tr>
<td><strong>Change</strong></td>
<td>Waves of change</td>
<td>Releases and iterations</td>
</tr>
</tbody>
</table>
Agenda

1. Introduction
2. Architecture
3. Agility
4. Building the Platform
5. Summary
Can you spot the innovation enablers?
Proven architecture practices

- Asset reuse
  - Survey Architecture Assets

- Decision capture
  - Document Architecture Decisions

- Component-based development
  - Outline Functional Elements
  - Outline Deployment Elements
  - Detail Functional Elements
  - Detail Deployment Elements

- Architecture proving
  - Verify Architecture
  - Build Architecture Proof-of-Concept

- Quality attribute-driven development

- Multiple views
  - Validate Architecture
  - Update Software Architecture Document
  - Review Architecture with Stakeholders
Agenda

1. Introduction
2. Architecture
3. Agility
4. Building the Platform
5. Summary
Manifesto for Agile Software Development

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 processes and tools
- **Working software** over comprehensive documentation
- **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.

“Software architecture has a history of excesses that in part spurred the reaction called Agile. Software architecture of the 1980s was famous for producing reams of documentation that no one read”. (Coplien & Bjørnvig, 2010)
## Architecture and agile – a clash of cultures?

<table>
<thead>
<tr>
<th>Architects’ Perception of Agile</th>
<th>Agilists’ Perception of Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leaves key decisions until last possible moment</td>
<td>Locks down key decisions too early</td>
</tr>
<tr>
<td>A lack of forethought leads to technical debt</td>
<td>BDUF over YAGNI 😊</td>
</tr>
<tr>
<td>A lack of modelling leads to significant rework when software is poorly thought through and does not scale</td>
<td>Comprehensive models slow your development efforts down to a snail’s pace</td>
</tr>
</tbody>
</table>
The voice of reason

- Architecture and agile are equally valid
  - “I see two parties not really understanding the real issues at hand, stopping at a very shallow, caricatural view of the “other culture”, not understanding enough of the surroundings, beliefs, values of the other one, and stopping very quickly at judging behaviors”. (Kruchten, 2010)

- Architecture and agile are complementary
  - “Scrum is a management and control process that cuts through complexity to focus on building software to meet business needs. Scrum is superimposed on top of and wraps existing engineering practices, development methodologies and standards”. (Schwaber & Beedle, 2002)
  - “Software architecture is part of product quality and isn’t tied to a particular process, technology, culture or tool”. (Nord & Tomayko, 2006)

- Architecture and agile need to be “balanced”
  - “The tension seems to lie on the axis of adaptation versus anticipation. Agile methods want to be resolutely adaptive: deciding at the “last responsible moment” for when changes occur. Agile methods perceive software architecture as pushing too hard on the anticipation side: planning too much in advance”. (Abrahamsson, Babar, & Kruchten, 2010)
Iterative development practices

- **Iterative development**
  - Incremental releases provide improvements in capability until the final system is complete

- **Risk-value lifecycle**
  - Phases represents the strategic plan for the project and drives the goals and objectives of each iteration

- **Shared Vision**
  - Ensures that all stakeholders share a common view of the problems being solved

- **Use Case-Driven Development**
  - Use cases make natural units of implementation in an iterative development approach

- **Release Planning**
  - Focused on the just-in-time project planning needed to scope the release of executable software within an iteration
Agile practices

- **Test-Driven Development (TDD)**
  - Creating tests that are a specification of what the code should do first

- **Continuous Integration**
  - Encourages frequent the integration and testing of programming changes

- **Refactoring**
  - Changing an existing body of code in order to improve its internal structure

- **Whole Team**
  - A focus on the value of highly-collaborative teams as exemplified by Scrum’s daily standup meeting. Instills of sense of collective ownership and responsibility

- **User Story-Driven Development**
  - Capture requirements in a lightweight manner. Encourages collaboration with the relevant stakeholders throughout a project

- **Team Change Management**
  - Supports the logging of defects or new requirements, by any member of the team, that are within the scope of the current iteration
Agile with discipline

Team size
Under 10 developers ↔ 1000's of developers

Compliance requirement
Low risk ↔ Critical, audited

Geographical distribution
Co-located ↔ Global

Domain Complexity
Straight-forward ↔ Intricate, emerging

Enterprise discipline
Project focus ↔ Enterprise focus

Organization distribution (outsourcing, partnerships)
Collaborative ↔ Contractual

Organizational complexity
Flexible ↔ Rigid

Technical complexity
Homogenous ↔ Heterogeneous, legacy
“Disciplined Agile Delivery” practices (examples)

- **Measured Performance**
  - This practice allows project and portfolio-level measurements to inform key business decisions

- **Formal Change Management**
  - Applied when change approval is required from stakeholders outside of the project team, or when a deliverable has been baselined as part of a contract and the deliverable needs to be modified

- **Concurrent Testing**
  - Often an independent test team is present (especially in larger organizations) that typically provide a level of user acceptance testing before the solution is put into production
# Practice Summary

## Traditional
- Multiple Views
- Quality Attribute-Driven Development
- Component-Based Development
- Asset Reuse
- Decision Capture
- Architecture Proving

## Iterative
- Iterative Development
- Risk-Value Lifecycle
- Shared Vision
- Use Case-Driven Development
- Release Planning

## Agile
- Test-Driven Development
- Continuous Integration
- Refactoring
- Whole Team
- User Story-Driven Development
- Team Change Management

## Disciplined Agile
- Measured Performance
- Formal Change Management
- Concurrent Testing
The “Business-IT” and “IT-IT” gaps

Business

Development

Operations

IT

Gap

Gap

Gap
Viewing the landscape as a supply chain

Business

Development

Operations

IT

Gap

Enterprise Architecture

Portfolio Management

Requirements

Design

Development

Test

Release

Monitoring
Technology imperatives

Collaborative Development
- Change management
- Software configuration management
- Continuous integration

Continuous Testing
- Quality management
- Test automation
- Service virtualization

Continuous Release
- Release management
- Environment provisioning
- Deployment automation

Continuous Monitoring and Optimization
- Application performance monitoring

Business

Enterprise Architecture
Portfolio Management
Requirements
Design
Development
Test
Release
Monitoring

Development

IT

MIND THE GAP
1. Define a lifecycle selection framework

<table>
<thead>
<tr>
<th>Management Influences</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Flexibility</strong></td>
</tr>
<tr>
<td>Management are willing to accept that business parameters, such as cost, schedule and intermediate milestones, are flexible</td>
</tr>
<tr>
<td><strong>Empowered Teams</strong></td>
</tr>
<tr>
<td>Management is willing to allow the team (including the product owner) to make key project decisions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stakeholder Influences</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Acceptance of Agile</strong></td>
</tr>
<tr>
<td>Stakeholders understand and accept agile practices and the consequences of following these</td>
</tr>
<tr>
<td><strong>Number of Stakeholders</strong></td>
</tr>
<tr>
<td>The number and diversity of stakeholder relationships to be managed is limited</td>
</tr>
<tr>
<td><strong>Stakeholder Responsiveness</strong></td>
</tr>
<tr>
<td>The business representative, end users and testers are committed to spending a good deal of time working with the team in an iterative fashion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project Team Influences</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Team skills</strong></td>
</tr>
<tr>
<td>Individuals on the team are team players, good communicators and are familiar with agile practices</td>
</tr>
<tr>
<td><strong>Embracing Change</strong></td>
</tr>
<tr>
<td>Team members expect and embrace frequent changes and iterative refinement of the solution</td>
</tr>
<tr>
<td><strong>Co-located Teams</strong></td>
</tr>
<tr>
<td>The project team will be co-located</td>
</tr>
<tr>
<td><strong>Team Stability</strong></td>
</tr>
<tr>
<td>Individuals will be assigned to the team for the duration of the project</td>
</tr>
<tr>
<td><strong>Team Roles</strong></td>
</tr>
<tr>
<td>Team members are able (and willing) to take on multiple roles during the project and to take on new roles if/when needed</td>
</tr>
<tr>
<td><strong>Agile Disciplines</strong></td>
</tr>
<tr>
<td>Team members have proven ability in performing disciplines that are critical for agile development with short iterations (design, testing and configuration management)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Technology Influences</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Development Environment</strong></td>
</tr>
<tr>
<td>The development environment (method, tools, training) will support an agile way of working (such as automated regression test, continuous integration and real-time dashboards) and is sufficiently mature</td>
</tr>
<tr>
<td><strong>Execution Environment</strong></td>
</tr>
<tr>
<td>The execution environment can support regular releases</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution Influences</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Requirements Churn</strong></td>
</tr>
<tr>
<td>There is a strong likelihood that there will be significant changes to requirements (and the solution) during the project</td>
</tr>
<tr>
<td><strong>Solution Complexity</strong></td>
</tr>
<tr>
<td>The required solution is relatively-complex (e.g. requires the use of unfamiliar technologies) and/or there are many different solution options</td>
</tr>
<tr>
<td><strong>Time-to-market</strong></td>
</tr>
<tr>
<td>The deadline (time) is the most important factor for the solution while the scope of the solution is flexible.</td>
</tr>
<tr>
<td><strong>Dependencies</strong></td>
</tr>
<tr>
<td>There are no (or only a few) dependencies on internal or external suppliers</td>
</tr>
<tr>
<td><strong>Release Frequency</strong></td>
</tr>
<tr>
<td>The solution can be subdivided in viable and meaningful business releases that can each be delivered within 3-4 months</td>
</tr>
<tr>
<td><strong>Demonstrability</strong></td>
</tr>
<tr>
<td>The solution can be easily demonstrated on an incremental basis (through a user interface, for example)</td>
</tr>
</tbody>
</table>
2. Tailor the method

<table>
<thead>
<tr>
<th>Element</th>
<th>Small Project</th>
<th>Large Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role</td>
<td>No specific architect role is required (architecture-related considerations will be handled by the team).</td>
<td>Different individuals are assigned to the following architecture-centric roles: Application Architect, Infrastructure Architect, Data Architect and Security Architect.</td>
</tr>
<tr>
<td>Task</td>
<td>An Architecture Overview is created as a sketch on a whiteboard and then photographed (it is not kept up to date).</td>
<td>An Architecture Overview is created as a UML model that is maintained.</td>
</tr>
<tr>
<td>Work product</td>
<td>Functional and Deployment viewpoints are used to communicate the architecture.</td>
<td>Requirements, Functional, Deployment, Validation, Performance and Security viewpoints are used to communicate the architecture.</td>
</tr>
</tbody>
</table>
3. Consider all elements of a development environment

- **Technology focus**
  - Method: Roles, work products, tasks, processes, standards, guidelines
  - Tools: Development tools & their integrations
  - Infrastructure: Locations, nodes & connectivity
  - Organization: Organizational roles & units
  - Adoption: Adoption plan, organizational change, metrics
  - Enablement: Training curriculum & courses
  - Cross-cutting Concerns: Functionality, qualities, constraints

- **People focus**
4. Adopt change incrementally

**Technical Scope (# practices)**

- **Organizational Scope (# users)**
  - Individual(s)
  - Project(s)
  - Organization

- **Small**
- **Moderate**
- **Complete**
5. Implement a center of excellence

Introducing new or improved capabilities results in a productivity dip.

One purpose of a CoE is to minimize this dip.

As well as reduce the time to reach a new steady state.
Summary

- A poor foundation can stifle innovation
- Architecture and agility are complementary innovation enablers
- With the right foundation, IT enables the enterprise
- A platform for innovation can be built practice by practice
- Building the platform is a transformational journey
Questions
Daily Apple TV giveaway

- Complete your session surveys online each day at a conference kiosk or on your Innovate 2013 Portal!
- Each day that you complete all of that day’s session surveys, your name will be entered to win the daily Apple TV!
- On Wednesday be sure to complete your full conference evaluation to receive your free conference t-shirt!
Acknowledgements and disclaimers

**Availability:** References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

© Copyright IBM Corporation 2013. All rights reserved.

- U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

IBM, the IBM logo, ibm.com, Rational, the Rational logo, Telelogic, the Telelogic logo, Green Hat, the Green Hat logo, and other IBM products and services are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml

Other company, product, or service names may be trademarks or service marks of others.
Thank You