Agile and Classical Project Management

George Merguerian
Senior and Managing Partner
Business Management Consultants

merguerian@bmc-online.com

Abstract

Agile methodologies enable the development and delivery of high quality software to customers and have been used with phenomenal success by thousands of companies for both internal applications development and the development of commercial software. This presentation will provide an overview of classical, agile and particularly of the RUP framework. It will also deal with the internal and external factors that need to be considered for their successful deployment.

Learning objectives

• Learn about the background of agile project management methodologies, their values and principles.
• Understand the key concepts of Iterative Software Development
• Understand the importance of Risk Driven approach to developing and implementing IS projects
• Learn about the RUP and SCRUM processes
• Understand what are the key success drivers of agile approaches
1. Historical
2. Why Agile
3. Agile Processes
4. RUP
5. SCRUM
6. Workshop
7. Conclusion
8. Q & A
Agile Methodologies

- ASD,
- Crystal Clear
- DSDM
- FDD
- RAD
- RUP
- SCRUM
- XP
- ….

Comparing Classical and Iterative PM

Low Ceremony
- Little Documentation/Light Process
- Lean, DSDM, XP, Crystal Clear, Orange, SCRUM

Manzo - Agile with Tek Code
- Boehm-Turner Risk Based

Hybrid

Manzo - Agile with Tek Code
- Boehm-Turner Risk Based

High Ceremony
- Plan-Driven, Well-Documented, Traceability, Change Control Board

Classical-
- SSDM

CMM

CMMI & ISO/IEC 15504

Iterative

Waterfall

RUP@EC

RUP
Incremental Development Proposal

The Spiral Model (1974)

The Spiral Model proposed by Barry W. Boehm

Agile Project Management

• In general applies for software development projects
• Approach is different to classical project management (waterfall)
Why Agile for Software projects?

- Users have difficulty in defining requirements upfront
- Agile allows requirements to filter in during the project lifecycle
- Agile allows customers to provide feedback throughout the project lifecycle.
- Frequent testing → Higher Quality software, less bugs, more stable system

Waterfall (classical) Approach

The project moves from:
Analysts → Designers → Integrators → Testers

→ ‘poorer’ responsibility for the final product.
Iterative Approaches

- Risk-driven
  - Starting with the technically riskiest elements of the project
- Client-driven
  - the client dictates the work for each iteration (business value driven).
- Each Iteration is Time-Boxed.
  - 2-5 weeks
  - Scope may change if goals are not achieved
- Each iteration builds on the previous one

All Agile Approaches Address Risks Early

Risk Reduction

Iterative Risk

Waterfall Risk
Agile Project Management

- Focuses on incremental development
  - An increment is termed as an iteration
  - Each iteration is like a small project with a fixed timeline

Each Iteration results into executable software

Iterative vs Waterfall

Waterfall approach

Iterative approach
Classical and Agile Project Management

CHAOS RESOLUTION BY STYLE

Shows classic CHAOS resolution results by waterfall versus the agile process from the CHAOS project database from 1994 to 2008.

Agile Methodologies

RUP & SCRUM
Classical IT Project Work Breakdown Structure

The Systems Development Lifecycle

- Requirements
  - RFP Document
- Feasibility
  - Feasibility Document
- Analysis
  - Current Physical Model
  - Current Logical Model
  - New System Logical Model
- Design
  - Detailed Design
- Construction
  - Software Code & Testing
  - Test Plan
- Deliver
  - Installation
- Verify
  - User Processes
- Maintenance
  - User Requirements
  - Logical Model
  - Design
  - Software Code
  - Test Plan

Agile Methodology RUP
RUP Process Framework

Process Structure: The basic elements of RUP
Inception phase is about ....

- Understanding the project requirement, scope and objectives and getting enough information to decide on how to proceed

### Disciplines
- Business Modelling
- Requirements
- Analysis & Design
- Implementation
- Testing
- Deployment
- Configuration & Change Mgmt
- Project Management
- Environment

### Phases
- Inception
- Transition
- Iteration
- Transition

### Example Iteration Workflow: Inception Phase

- Project Manager
- Business Case
- Conceptual business case
- Detailed business case
- Project plan
- Project plan approved
- Software requirements definition
- Software requirements identified
- Architecture identified
- Design and testing
- Test results
- Iteration plan
- Iteration feedback
1. Inception – Understanding what to build

Produces a vision document:
1. Who needs the project
   - Who will use it
   - The benefits of the project, what problem it will solve..
   - Key deliverables, Key features (use cases)
2. The benefits that the application will produce – business-wise, the problems it will solve, risks
3. Who are the key stakeholders
4. What will the product do – key use cases
5. Non functional requirements – database support, OS related issues, scaleability etc.

Example of a Vision document

1. Problem Statement

1.1. Problem Statement
The problem of Preparing and making readily available the information and produced by the evaluation system of the Commission.
Affects Evaluation functions, programme and policy-makers, managers, and produced by the evaluation system of the Commission.

1. Problem Statement

1.1. Problem Statement
The problem of Preparing and making readily available the information and produced by the evaluation system of the Commission.
Affects Evaluation functions, programme and policy-makers, managers, and produced by the evaluation system of the Commission.

2. Proposed solution

3. Proposed Approach

5. User Environment

7. Features – Need, Priority, Feature

8. Planned Resources, Constraints (security, other)
A simplified example of a vision document

1. Problem: Gary's team is unable to capture effort spent on various software development projects. As a result cannot monitor progress against estimates. This makes it difficult to charge customers correctly, and estimate effort required for future projects.

2. Vision statement: A software tool that measures time spent, gathers data and allows its extraction solving the above problems.

3. Stakeholders: Gary's team, our organisation which is specialised in PM software, Administration people in Gary’s organisation

4. Use Cases
   - Measure time for an activity
   - Extract weekly time sheets
   - Consolidate data for project
   - Set up tool and database for project

In RUP Requirements Captured Through Use Cases

- A use case model describes a system's functional requirements in terms of actors and use cases.
- A use case describes how each actor will interact with the system and derive value from it. It describes the functional requirements in a manner that is understandable to the customer.
- An actor represents a type of user of the system, or another system that will interact with the system.
How to develop Use Cases?

1. Identify all possible actors — write a brief description about them (1 or 2 sentences)
2. Associate each actor with use cases. Capture the actor's interactions with the system by providing a brief description of the use case.
3. Determine whether each use case requires interaction with other users or systems.
4. Identify the most essential or critical use cases.

<table>
<thead>
<tr>
<th>Actor</th>
<th>Goal</th>
<th>Brief</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Staff</td>
<td>Modify the admin area lattice</td>
<td>Staff adds metadata (admin hierarchy, currency, language code, street types, etc.) to reference database. Contact info of source data is catalogued. This is a special case of updating data</td>
</tr>
</tbody>
</table>

Use Case Example - written format

- Example of a Use Case description: Somebody paying for an advertisement using credit card
  - Use Case Title: Pay for a job posting
  - Primary actor: Recruiter
  - Successful Outcome: Job is posted; recruiter’s credit card is charged
  - Main Success Scenario:
    1. Recruiter submits credit card number, date, and authentication information.
    2. System validates credit card.
    3. System charges credit card full amount.
    4. Job posting is made viewable to Job Seekers.
    5. Recruiter is given a unique confirmation number.
  - Extensions:
    2a: The card is not of a type accepted by the system:
      2a1: The system notifies the user to use a different card.
    2b: The card is expired:
      2b1: The system notifies the user to use a different card.
    2c: The card number is invalid:
      2c1: The system notifies the user to re-enter the number.
    3a: The card has insufficient available credit to post the ad:
      3a1: The system charges as much as it can to the current credit card.
      3a2: The user is told about the problem and asked to enter a second credit card for the remaining charge. The use case continues at Step 2.
Use Case Examples – UML

Use Case Model is used to create the interrelated models
Elaboration
Key Objectives

1. Get a more detailed understanding of requirements
2. **Design, validate and baseline the architecture**
3. Mitigate risks and better determine schedule and budget estimates
4. Refine the development case and put the development environment into place

---

Construction
Key Objectives

1. Minimise development costs by optimising resources and avoiding unnecessary rework
2. Achieve pre-determined quality levels rapidly
3. Achieve useful versions rapidly
4. Complete the analysis, design, development and testing of all required functionality.
5. Complete the development of the product for transition to users
Transition
Key Objectives

1. Beta Test to validate that the user expectations are met
2. Train users and maintainers to achieve user self-reliability
3. Prepare deployment site and convert operational databases
4. Prepare for launching, Communications …
5. Agree with Stakeholders that you have delivered what was asked for as per the evaluation criteria of the Vision document
6. Record lessons learned

Agile Methodology
SCRUM
Scrum Process

• Each Sprint is 30 calendar days
• Each Sprint is preceded by a planning meeting to decide what will be done in the next Sprint
• At the end of Sprint – Review meetings are held
• The desired outputs are put in a Product Backlog (Requirements)
• Daily Scrums of 15 minutes to synchronise work – what was done, what will you do now and next Scrum
SCRUM Artefacts

- Product Backlog – Requirements for each Sprint
- Sprint Backlog - the tasks that the Team defines to convert the Product Backlog into shippable functionality
- Tasks – 4-16 hours work
- Software Releases
- Burndown Chart showing remaining work
Product Backlog Example

### Weather on Mobile

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Sprint 1</th>
<th>Sprint 2</th>
<th>Sprint 3</th>
<th>Sprint 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Display current temperature in a graph and sample temperature test</td>
<td>13</td>
<td>8</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Implement support for several days</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Implement support for rain, snow, etc. icons</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Add weather warning</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Fetch weather data from the weather provider system</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Add the weather forecast, snow, etc. details from the provider</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Add support for rain, snow, etc. icons</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>Add support for several days</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Add support for weather warning</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Sprint 1
- **Focus is scope that can be developed in time given**

### Sprint 2
- **Focus is scope that can be developed in time given**

### Sprint 3
- **Focus is scope that can be developed in time given**

### Sprint 4
- **Focus is scope that can be developed in time given**
### SCRUM Product Backlog

**Source:** Luc Segers

#### Progress for Iteration

<table>
<thead>
<tr>
<th>Task</th>
<th>Name</th>
<th>Description</th>
<th>Effort</th>
<th>Date</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Sprint Backlog

**Source:** Luc Segers
### Sprint Backlog Example

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
<th>Est</th>
<th>By</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Finish database versioning</td>
<td>16</td>
<td>KH</td>
</tr>
<tr>
<td>2</td>
<td>Get rid of unneeded shared Java in database</td>
<td>8</td>
<td>KH</td>
</tr>
<tr>
<td>3</td>
<td>Add licensing</td>
<td>16</td>
<td>TG</td>
</tr>
<tr>
<td>4</td>
<td>Demo / Eval licensing</td>
<td>16</td>
<td>TG</td>
</tr>
<tr>
<td>5</td>
<td>Update Manager</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>Fix formats we support are out of date</td>
<td>160</td>
<td>TG</td>
</tr>
<tr>
<td>7</td>
<td>Round Out Analyses</td>
<td>250</td>
<td>MC</td>
</tr>
</tbody>
</table>

### Sprint Backlog - Remaining Hours Example

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Mon</th>
<th>Tues</th>
<th>Wed</th>
<th>Thurs</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code the user interface</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code the middle tier</td>
<td>16</td>
<td>12</td>
<td>10</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Test the middle tier</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>Write online help</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write the foo class</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Add error logging</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>
Scrum Roles

1. **Product Owner** – responsible for the resulting system; uses Product Backlog to ensure most useful functionalities are addressed
2. **Team** – develops functionality
3. **Scrum Master** – responsible for scrum process; its implementation in the organisation and for teaching scrum

People in the project have authority, others cannot interfere
A Director in a company called Luminous faces the following situation:

- His 6 managers deliver on a weekly basis the status of 200 activities and projects
- The Director spends over a day per week at reading reports
- The Director requests that the IT department of Luminous develop a "Dashboard" type of reporting system that at a glance gives a graphical representation of the status of the key 20 projects
The Director’s Requirements

1. An easy to use Dashboard
2. Weekly Updates of Dashboard
3. Show status of at least 20 projects
4. Links from the dashboard to the project files
5. A database of the Dashboard reports
6. The Dashboard should be accessible via the web

Workshop I – RUP

Assume that you are stepping in as the consultant to develop this application
- Write a few client requirements
- Write one Use Case OR Make a UML type diagram for a Requirement(s)
- List 3-4 Business Risks

For the Workshop Assign:
- A Project Manager who facilitates the workshop and keeps time;
- A Reporter who writes and presents the workshop’s outputs (in cooperation with team members).
Workshop I – SCRUM

Assume that you are stepping in as the consultant to develop this application

• Develop a Product Backlog for SPRINT 1
• Develop a SPRINT Backlog for a Requirement(s)
• List 3-4 Business Risks

For the Workshop Assign:

• A SCRUM Master who facilitates the workshop and keeps time;
• A Reporter who writes and presents the workshop’s outputs (in cooperation with team members).

In Agile Approaches

• Emphasis is on
  – Communications and interaction
  – Resource competency and involvement
Agile is not just a methodology

The most important thing to know about Agile methods is that there are only Agile teams. Agile processes are environments for a team to learn how to be Agile.

While a new process can easily improve team productivity by a fraction, enabling the team to work effectively as a cohesive unit can improve productivity by several times.

Instead of managing our activities we will manage requirements and demonstrate each new version to the customer.

Modified from: Agile-processes.org

Conclusion

Iterative approach advantages

- Easier to meet changing requirements (main cause of trouble for IT/IS projects)
- Integration becomes easier (not the big thing at the end)
- Risks are discovered earlier
- Management can make changes to the product
- Team members gain expertise as they assume several roles during each development cycle/iteration.