Heavy vs Light Methodologies: Bulimic or Anorexic?

Fernando Brito e Abreu (fba@di.fct.unl.pt)
Universidade Nova de Lisboa (http://www.unl.pt)
QUASAR Research Group (http://ctp.di.fct.unl.pt/QUASAR)

Abstract

- From anorexic to bulimic
- Overview of heavy-weight methodologies
- Origins of light-weight methodologies
- The “Manifesto”
- Agility example: XP
- The dark side of the light
- Planned (RUP) vs Agile (XP)
- When to be agile?
Some hype ...

<table>
<thead>
<tr>
<th>Plan-driven methodologies</th>
<th>Agile software development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heavy-weight methodologies</td>
<td>Light-weight methodologies</td>
</tr>
<tr>
<td>CMM, ISO9000-3, ISO12207, PSP, TSP, ISO15504 (SPICE), RUP, CMMi, ...</td>
<td>Extreme Programming (XP), Scrum, Feature-Driven Development (FDD), Adaptive Software Process, Crystal Light Methodologies, Dynamic Systems Development Method (DSDM), Lean Development</td>
</tr>
<tr>
<td>Fat? Bulimic?</td>
<td>Thin? Anorexic?</td>
</tr>
</tbody>
</table>

From anorexic to bulimic

“If you don’t know where you are going, every road will take you there...”

Alice in Wonderland

- Many projects run unconstrained
  - Total freedom for artists 😊
  - Unpredictable results 😊
- DoD Mil. Std 2167A
  - Highly constrained, high organizing overheads 😊
  - Very well defined deliverables 😊
Overview of heavy-weight methodologies

- Processes and tools
- Comprehensive documentation
- Contract negotiation
- Following a plan

“On projects with more than 250 people, methodology will have almost no impact on success or failure – politics will dominate.”

Jim Highsmith

CMM - Capability Maturity Model

- Initial (1)
- Repeatable (2)
- Defined (3)
- Managed (4)
- Optimising (5)

Progress
CMM - Capability Maturity Model

Quality management  Process measurement and analysis

Initial (1)
- Software configuration management
- Software quality assurance
- Software subcontract management
- Software project tracking and oversight
- Software project planning
- Requirements management

Repeatable (2)
- Peer reviews
- Intergroup coordination
- Software product engineering
- Integrated software management
- Organization process definition
- Organization process focus

Defined (3)
- Process change management
- Technology change management
- Defect prevention
- Optimizing (5)
- Software quality management
- Quantitative process management

Managed (4)
- Process change management
- Technology change management
- Defect prevention

Process Advisor
(Roger Pressman & Associates)
ISO 15504

• SPICE - Software Process Improvement and Capability dEtermination
• Initiative of WG10 (Process Assessment) - ISO/IEC JTC1/SC7 (Software Engineering)

OBJECTIVES:
- Unify software process assessment efforts
- Elaboration of a set of international standards

ISO 15504
Organizations involved

Australian Software Quality Research Institute
Bell Canada
Northern Telecom
Bell Northern Research (BNR)
BOOTSTRAP Consortium
British Telecommunications Plc.
Centre de Recherche d'Informatique de Montréal

Defense Research Agency, UK
European Software Institute
Software Engineering Institute
Etnoteam, Italy
University of Oulu, Finland
Bellcore, EUA

... and other organizations from Japan, South Africa, France, Ireland, Spain, ...
ISO 15504

Considers 5 Generic Process Categories:

- **CUS** - customer-supplier process category
- **ENG** - engineering process category
- **PRO** - project management process category
- **SUP** - support process category
- **ORG** - organization process category

Capability Maturity Model Integrated (CMMI®)

- Is an integrated model to propel process improvements in systems engineering and software engineering.
- The model encompasses:
  - 5 maturity levels
  - 25 Process Areas (PAs)
  - Several flavors (SE/SW/IPPD/SS)
**Capability Maturity Model Integrated (CMMI®)**

- **Level 2** - Requirements Management, Project Monitoring and Control, Project Planning, Supplier Agreement Management, Configuration Management, Process & Product QM, Measurement & Analysis
- **Level 3** - Requirements Development, Technical Solution, Product Integration, Organizational Training, Verification Validation, Risk Management, Decision Analysis & Resolution, Integrated Project Management, Organizational Process Focus, Organizational Process Definition
- **Level 4** - Quantitative Project Management, Organizational Process Performance
- **Level 5** - Organizational Innovation & Deployment, Causal Analysis & Resolution
- Additional Requirements of IPPD - Changes to Integrated Project Management, Integrated Teaming and Organizational Environment for Integration
- Additional Requirements of SS - Integrated Supplier Management

**Rational Unified Process (RUP)**

<table>
<thead>
<tr>
<th>Year</th>
<th>Description</th>
<th>Version</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1998</td>
<td>Requirements College</td>
<td>SQA Process</td>
<td>Rational Unified Process 5.0</td>
</tr>
<tr>
<td>1997</td>
<td>OMT approach</td>
<td>UML 1.0</td>
<td>Rational Objectory Process 4.1</td>
</tr>
<tr>
<td>1995</td>
<td>Rational Approach</td>
<td>OOSE</td>
<td>Rational Objectory Process 4.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Objectory Process 3.8</td>
</tr>
</tbody>
</table>
Rational Unified Process (RUP)

- Use-Case Model
  - Specified by
    - Analysis Model
      - Realized by
        - Design Model
          - Distributed by
            - Deployment Model
              - Implemented by
                - Implementation Model
                  - Verified by
                    - Test Model

Rational Unified Process (RUP)

- Role (Who)
  - Use-Case Designer
  - Use-Case Analysis
  - Use-Case Design

- Activities (How)
  - Use-Case Realization

- (What)
  - Responsible for

- Workflow (When)
  - (Who)
  - (What)
  - (How)
Origins of light-weight methodologies

- Agility
  - The ability to both create and respond to change in order to profit in a turbulent business environment
    - Companies need to determine the amount of agility they need to be competitive

- Chaordic (ex: agile view)
  - Exhibiting properties of both chaos and order
    - The blend of chaos and order inherent in the external environment and in people themselves, argues against the prevailing wisdom about predictability and planning
    - Things get done because people adapt, not because they slavishly follow processes
  - An agile view is a chaordic view
The Agile Manifesto Subscribers

Alistair Cockburn  Jon Kern
Andrew Hunt  Ken Schwaber
Arie van Bennekum  Kent Beck
Brian Marick  Martin Fowler
Dave Thomas  Mike Beedle
James Grenning  Robert C. Martin
Jeff Sutherland  Ron Jeffries
Jim Highsmith  Steve Mellor
Ward Cunningham  

The Agile Manifesto [Feb 2001]

“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 on the items on the right, we value the items on the left more.”
Agile Management Issues

• Promote teambuilding and trust
• Set an open tone with the customer(s) organizations
• Interpret and translate risks for overall program integration and a common view
• Have a robust, flexible, and adaptable configuration and data management system

Summary of Agile Characteristics

• Adaptability rather than predictability
• People rather than development process
  - Being agile means accepting that outcomes are not predictable and that processes are not repeatable
• Collaborative values and principles
• A barely sufficient methodology
  - “the conventions we agree to”
  - Processes are described in manuals; practices are what happen in reality
Agility example: XP

Extreme Programming Project

User Stories
- Requirements
- System Metaphor
- Uncertain Estimates
- Spike

Architectural Spike
- Release Planning
- Confident Estimates
- Next Iteration

Test Scenarios
- New User Story
- Project Velocity
- Bugs

Iteration
- Release Plan
- Latest Version
- Customer Approval

Acceptance Tests
- Small Releases

XP – Practices

- The Planning Game
- Small Releases
- System Metaphor
- Simple Design
- Testing
- Refactoring

- Pair Programming
- Collective Ownership
- Continuous Integration
- 40-hour week
- On-site-Customer
- Coding Standards
**XP - Schedule ...**

XP Planning/Feedback Loops

The dark side of the light ...

"XP Considered Harmful ... for Reliable SW Development"

[Gerold Keefer, 2002]

- The embrace change value ...
- The practice of refactoring ...
- The simplicity value ...
- The practice of pair programming ...

...
Planned (RUP) vs Agile (XP)

RUP
- Business Modeling
- Req. Management
- Analysis & Design
- Implementation
- Test
- Deployment
- CCM
- Project Management
- Environment

XP
- Coding
- Testing
- Listening
- Designing

Process Disciplines
- Business Modeling
- Requirements
- Analysis and Design
- Implementation
- Test
- Deployment
- Supporting Disciplines
- Configuration Mgmt
- Management
- Environment

Phases
- Inception
- Elaboration
- Construction
- Transition

XP-Window

Iterations
- Iteration(s)
- Iteration 1
- Iteration 2
- Iteration n-1
- Iteration n-2
- Iteration n
- Iteration n+1
When to be agile?

- Problems characterized by change, speed, and turbulence are best solved by agility.
  - Accelerated time schedule combined with significant risk and uncertainty that generate constant change during the project.

- Is your project more like drilling for oil or like managing a production line?
  - Oil exploration projects need Agile processes.
  - Production-line projects are often well-served by rigorous methodologies.

That’s all folks 😊

Fernando Brito e Abreu
fba@di.fct.unl.pt
http://ctp.di.fct.unl.pt/QUASAR

- Questions?
References

Agile Software Development Ecosystems, Jim Highsmith
Agile Modeling: Effective Practices for Extreme Programming and the Unified Process, Scott Ambler
Agile Development, Rich McCabe, May 2003
Agile Software Development with Scrum, Ken Schwaber and Mike Beedle, Prentice-Hall Inc., 2002
Extreme Programming Explained: Embrace Change, Kent Beck
Refactoring: Improving the Design of Existing Code, Martin Fowler

A Practical Guide to Feature-Driven Development, Stephen Palmer and John Felsing
Extreme Programming Explained, Kent Beck, Addison Wesley, 2000
Refactoring: Improving the Design of Existing Code, Martin Fowler et al., Addison-Wesley, 2000
Extreme Programming Installed, Ron Jeffries, Ann Anderson, Addison-Wesley