Beyond Agility

LARS MATHIASSEN
Center for Process Innovation
J Mack Robinson College of Business
Georgia State University
The 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 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.
Competing values
(Quinn & Rohrbaugh, 1983)

+ Organic, rather than bureaucratic software development
  + External, rather than internal orientation
    + Ends, rather than means emphasis

- How do we balance requirements and resources?
- How do we document and improve productivity?
  - How do we ensure software innovation?
How do we balance requirements and resources?

Repeat-ability & Response-ability

(Napier, Mathiassen & Johnson, 2006)

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Repeat-ability</th>
<th>Response-ability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nature of requirements</strong></td>
<td>• Requirements represent software capabilities</td>
<td>• Requirements are perceptions of software capabilities</td>
</tr>
<tr>
<td></td>
<td>• Requirements are explicated as texts in documents</td>
<td>• Requirements are tacitly embedded in social relationships</td>
</tr>
<tr>
<td><strong>Requirements capture</strong></td>
<td>• Requirements are derived through specification</td>
<td>• Requirements are discovered through negotiation</td>
</tr>
<tr>
<td></td>
<td>• Interaction is formal</td>
<td>• Interaction is informal</td>
</tr>
<tr>
<td><strong>Requirements usage</strong></td>
<td>• Requirements are baselined and predate development</td>
<td>• Requirements emerge through development</td>
</tr>
<tr>
<td></td>
<td>• Requirements are stored with traceability to source code</td>
<td>• Requirements are expressed through software solutions</td>
</tr>
<tr>
<td><strong>Change management</strong></td>
<td>• Requirements changes are exceptions and must be managed</td>
<td>• Requirements changes are expected and must be embraced</td>
</tr>
<tr>
<td><strong>Improvement goal</strong></td>
<td>• The goal is to reduce process variance through best practices</td>
<td>• The goal is to increase customer satisfaction through collaboration</td>
</tr>
</tbody>
</table>
How do we document and improve productivity?

Metrics & Media

(Frederiksen & Mathiassen, 2008)

- Information-centric approach
- Driven by engineering and management needs for information
- Intended meaning (sender) and interpreted meaning (receiver)

- Reflecting: information-enabled retrospectives
- Responding: information-enabled requests
How do we ensure software innovation?

The learning imperative
(March, 1991)

In a constantly changing world, learning is key to business success

There are two basic forms of organizational learning
• Exploitation of old certainties
• Exploration of new possibilities

Managers must balance these two forms of learning

Firm level coordination therefore involves contradictory choices
Organizational ambidexterity

(Tushman & O’Reilly, 1996)

Ambidextrous organizations compete by simultaneously
• optimizing efficiency, cost, and incremental innovation
• exhibiting flexibility, speed, and radical innovation

Structural ambidexterity

Contextual ambidexterity
When context is king
(Gibson & Birkinshaw, 2004)

Appropriate context empowers individuals to exploit and explore

High-performing contexts are characterized
- **Performance management (discipline & stretch)**
- **Social support (support & trust)**

<table>
<thead>
<tr>
<th>Social Support</th>
<th>Performance management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Low Performance</td>
</tr>
<tr>
<td>High</td>
<td>Country Club</td>
</tr>
<tr>
<td></td>
<td>High Performance</td>
</tr>
</tbody>
</table>

- **Low Performance**
- **Burnout context**
Building contextual ambidexterity

(Napier, Mathiassen & Robey, 2011)

Context

*Performance Mgt.*
- Discipline
- Stretch

*Social Support*
- Support
- Trust

Process

Diagnosing
Visioning
Intervening
Practicing

Content

Alignment
Adaptability
Action research at *TelSoft*
(Napier, Mathiassen & Robey, 2011)

- Map-based IT services, targeting telecommunications industry
- Two dominating customers, many small customers
- Failed investment in “speculative development”
Key challenges

Developing project KPIs
Creating and present market analyses
Responding to account pressures
Assessing account options
Designing and managing innovation initiatives
Prioritizing and allocating resources
Contextual outcomes

- Low Social Support
  - Low Performance Context
  - Country Club Context
  - Phase 1
  - Phase 2

- High Social Support
  - High Performance Context
  - Burnout Context
  - Phase 3
  - Phase 4
Ambidexterity outcomes

ALIGNMENT

Low

High

ADAPTABILITY

Low

High

Exploitation Focused

Phase 2

Phase 3

Phase 4

Ambidextrous

Reactive

Phase 1

Exploration Focused

Phase 2

Phase 3

Phase 4
Managerial principles

- **Principle 1 (Process):** Establish a software coordination group with an agenda focusing on product portfolio management, project portfolio management, market and technology research, and software process improvement.

- **Principle 2 (Process):** Guided by a software charter, develop a context that facilitates and enables ambidextrous capability through an iterative process of diagnosing, establishing, acting, and learning.

- **Principle 3 (Content):** Increase alignment by focusing on project portfolio management and software process improvement.

- **Principle 4 (Content):** Increase adaptability by focusing on product portfolio management and market & technology research.

- **Principle 5 (Context):** Apply established principles for contextual ambidexterity to improve performance management and social support.
Epilogue

“The bad news is that we will never find a process that allows us to design software in a perfectly rational way. The good news is that we can fake it. We can present our system to others as if we had been rational designers and it pays to do so.”

(Parnas & Clements, 1986)
Readings


