Strategic Application of Software Development Process for Business Oriented Projects

Glenn Engstrand
President and Founder
Dynamical Software, Inc.
Project Failure Studies

Robbins-Gioia Survey

Conference Board Survey

KPMG Canada Survey

CHAOS Report

OASIG Survey
ACM Queue

1 Minute Risk Assessment Tool

Use of Inappropriate Methodology

Lack of Customer Involvement

Lack of Formal Project Management Practices

Dissimilarity to Previous Projects

Project Complexity

Requirements Volatility
Requirements Churn

Overly Optimistic Estimates

Rush to make deadlines increases technical debt

Most Significant Phase is Feature Enhancement Maintenance

Quality Control

Peer Review

White box testing
Why Projects Fail

Information Overload

A new methodology is overly hyped resulting in high expectations with inappropriate and misplaced enthusiasm.

Just Do It

An obsessive focus on meeting short term milestones and budgetary goals results in excessive technical debt resulting in rigidity over time.
Common Theme

Lack of understanding as to how to apply fundamentally sound academic theory to practice in a business context.

The only thing that the customer is willing to directly pay for is code.
Information Overload

Theory

Popular
Over hyped
Complex

Initial Enthusiasm

Apply Everything
Nerds Competing

Eventual Disappointment

Too much overhead
Burn Out
Discard Everything
Examples

Computer Aided Software Engineering

Rational Unified Process

eXtreme Programming

Agile Methodology

SCRUM
Rational Unified Process

- Process Workflows
  - Business Modeling
  - Requirements
  - Analysis and Design
  - Implementation
  - Test
  - Deployment

- Supporting Workflows
  - Configuration and Change Management
  - Project Management
  - Environment

Inception | Elaboration | Construction | Transition
---|---|---|---
preliminary iteration(s) | iter #1 | iter #2 | iter #n | iter #n+1 | iter #n+2 | iter #m | iter #m+1
eXtreme Programming

Extreme Programming Project

User Stories
- Requirements
- New User Story
- Project Velocity
- Test Scenarios

Architectural Spike
- System Metaphor
- Uncertain Estimates
- Confident Estimates
- Next Iteration

Release Planning
- Release Plan
- Latest Version

Iteration
- Bugs

Acceptance Tests
- Customer Approval

Small Releases

Copyright 2000 J. Donvan Wells
Agile

- customer involvement
- embrace change
- short iteration cycles
- no silos
- motivation
- face time

- working software
- sustainable
- peer reviewed design
- simplicity
- self organizing
- optimizing
SCRUM

Scrum: 15 minute daily meeting. Team member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?

Sprint Backlog: Feature(s) assigned to sprint
Backlog items expanded by team

Product Backlog: Prioritized product features desired by the customer

New functionality is demonstrated at end of sprint
Solution

Cafeteria Approach

- Adopt only what makes sense
- Don't try to apply everything
- Don't change corporate process just to learn something

Expectation Management

- How Adoption Affects Productivity
  - Initially Lower
  - Eventually Higher
Just Do It

The Pendulum Swings

See Previous Scenario

Penny Wise and Pound Foolish

Developers
- Under Educated
- Inexperienced
- Cheap

Managers
- Missed Milestones
- Time Micro Management
- Short Term Thinking
Solution

Focus on the Total Cost of Ownership

What makes some bugs expensive and others cheap?

- The earlier in the SDLC phase
- The longer it takes to fix
- The higher the cost of the bug
- Technical Debt
Capability Maturity

- Capability Maturity Model
- Software Engineering Institute
- Carnegie Mellon University
- Originally a USAF funded study
- Managing the Software Process
- Not just for software
- Now called CMMI (Integration)
<table>
<thead>
<tr>
<th>Capability Maturity</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>heroics</td>
</tr>
<tr>
<td>Managed</td>
<td>improve quality</td>
</tr>
<tr>
<td>Defined</td>
<td>formal process</td>
</tr>
<tr>
<td>Quantitative</td>
<td>taking measurements</td>
</tr>
<tr>
<td>Optimizing</td>
<td>track and improve</td>
</tr>
</tbody>
</table>
**Look For Pain Points**

<table>
<thead>
<tr>
<th>Initial</th>
<th>burn out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managed</td>
<td>high bug counts</td>
</tr>
<tr>
<td>Defined</td>
<td>reign in the rogues</td>
</tr>
<tr>
<td>Quantitative</td>
<td>your work is done here</td>
</tr>
<tr>
<td>Optimizing</td>
<td></td>
</tr>
</tbody>
</table>
Advocating Process

• Introducing process into a business setting is a sales job.
• Expect resistance because of the perception that all that the customer is buying is code.
• You must get buy-in from management.
• You must also get buy-in from engineering.
## Selling to Management

<table>
<thead>
<tr>
<th>Deliverables</th>
<th>Talking Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>user stories, walk-throughs, story boards</td>
<td>gain deep insight and understanding of your market</td>
</tr>
<tr>
<td>use cases</td>
<td>prioritizes features</td>
</tr>
<tr>
<td>UML</td>
<td>remove mistakes early, mentoring, reduce technical debt</td>
</tr>
<tr>
<td>FPA</td>
<td>formalized approach to estimates</td>
</tr>
<tr>
<td>gantt charts</td>
<td>time and resource management</td>
</tr>
<tr>
<td>burn-down charts</td>
<td>know where you are in the project</td>
</tr>
</tbody>
</table>
## Selling to Management

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Talking Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>information architecture</td>
<td>customer relevance</td>
</tr>
<tr>
<td>eXtreme Programming</td>
<td>good for when you have junior coders</td>
</tr>
<tr>
<td>Agile</td>
<td>keeps the team focused and learning what the customer wants</td>
</tr>
<tr>
<td>SCRUM</td>
<td>increases team motivation</td>
</tr>
</tbody>
</table>
## Selling to Engineering

<table>
<thead>
<tr>
<th>Deliverables</th>
<th>Talking Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>user stories, walk-throughs,</td>
<td>assess requirements quality before committing to</td>
</tr>
<tr>
<td>story boards</td>
<td>estimates</td>
</tr>
<tr>
<td>use cases</td>
<td>technical insight to requirements</td>
</tr>
<tr>
<td>UML</td>
<td>software quality</td>
</tr>
<tr>
<td>FPA</td>
<td>takes estimating off your plate</td>
</tr>
<tr>
<td>gantt charts</td>
<td>fewer distractions from management</td>
</tr>
<tr>
<td>burn-down charts</td>
<td>permitted to revise estimates</td>
</tr>
</tbody>
</table>
## Selling to Engineering

<table>
<thead>
<tr>
<th>Methodology</th>
<th>Talking Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>information architecture</td>
<td>coders hate to come up with labels</td>
</tr>
<tr>
<td>eXtreme Programming</td>
<td>coders hate to document</td>
</tr>
<tr>
<td>Agile</td>
<td>get to complete on your promises</td>
</tr>
<tr>
<td>SCRUM</td>
<td>flattering to be served by management</td>
</tr>
</tbody>
</table>
Code Roller™

• collaborative software development project life cycle management solution
• community where entrepreneurs and engineers get together to produce great software
• raise the level of intelligent awareness of business savvy SDLC process
• free to use
Code Roller™

- Collaborative
- Life cycle
- Intelligent
- Cybernetic
- Knowledge based

- Software
- Product
- Project
- Management
Code Roller™

- requirements management
- change management
- configuration management
- release management

- records management
- compliance management
- content management
Thank You

http://code-roller.com

http://www.dynamicalsoftware.com

glenn@dynamicalsoftware.com