CMMI and Agile

Tom Cagley
Vice President, Consulting Services
Agile and CMMI

Why
Or
Why Not?
Common Sense and Process Discipline

<table>
<thead>
<tr>
<th>Process Discipline</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality</td>
<td>Assembly Line?</td>
<td>Unfettered Agile?</td>
</tr>
<tr>
<td>Mindless bureaucracy</td>
<td>Death by Paper CMMI?</td>
<td>Mindless chaos</td>
</tr>
<tr>
<td>Creative chaos</td>
<td>Ad Hoc?</td>
<td></td>
</tr>
</tbody>
</table>

Adapted from: Sanjiv Ahuja, former COO of Telcordia
# Work and Control

<table>
<thead>
<tr>
<th>Output Measures</th>
<th>Production</th>
<th>Craft</th>
<th>Information Processing</th>
<th>Knowledge Worker</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Volume &amp; Quality</td>
<td>Quality &amp; Value</td>
<td>Volume &amp; Quality</td>
<td>Quality &amp; Value</td>
</tr>
<tr>
<td>Control Mechanisms</td>
<td>Compliance to procedures and standards</td>
<td>Adherence to good practice (professionalism)</td>
<td>Compliance to procedures and standards</td>
<td>Adherence to good practice (professionalism)</td>
</tr>
<tr>
<td>Control Level Focus</td>
<td>Management</td>
<td>Worker</td>
<td>Management</td>
<td>Worker</td>
</tr>
</tbody>
</table>
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

While there is value in the items on the right, we value the items on the left more.
Twelve Principles, Part 1

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
Twelve Principles, Part 2

• Working software is the primary measure of progress.
• Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.
• Continuous attention to technical excellence and good design enhances agility.
• Simplicity – the art of maximizing the amount of work not done – is essential.
• The best architectures, requirement, and designs emerge from self-organizing teams.
• At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
Two Sides Of Agile

- TDD
- FDD
- AUP
- xP
- Project Management
- Technical
- Crystal
- APPM
- Scrum
Agile Techniques

- Project Management
  - Iterations
  - Sprint Planning
  - Retrospectives
  - Standup Meetings
  - Release Planning
  - Product Planning
  - Definition of Done

- Technical
  - Automated Testing
  - Pair Programming
  - Continuous Integration
  - Definition of Done
  - Standards
  - Architecture
  - Continuous Improvement

Just doing one does not truly make you Agile!
What Agile Manifesto Doesn’t Mean

• Ad hoc

• Code and run

• No documentation

• No discipline

• No process

• No planning

• Just Scrum
CMMI: Maturity Levels

- Requirements, processes, work products and services are managed. The status of the work products and the delivery of services are visible to management at defined points.
- Commitments are established among relevant stakeholders and are revised as needed. Work products are reviewed with stakeholders and are controlled.
- The work products and services satisfy their specified requirements, standards and objectives.
What CMMI Is and Is Not

• Is a model for organizational improvement
  – Benchmarks the process, not the product
  – Provides a framework for staged growth, i.e. process and capability maturity
  – Continuous process improvement stressed

• Is an underlying structure for reliable and consistent CMMI-based appraisals

• Is a community-owned guide; the result of a national consensus
  – Use of standardized language and goals facilitates adopting best practices from across the software community

• Is not an academic “ivory tower” exercise

• Is not limiting
  – Does not limit the choice of life cycle
  – Does not require or preclude specific software, technologies or language
  – Does not require that documentation conform to any particular set of standards
Agile and CMMI Assumptions

**Agile**
- Internal customer
- Dynamic environment
- Business system
- Low cost of early failure
- High trust
- Time & material
- Programming is a craft
- Internalised plans, qualitative control
- Tacit knowledge
- Refactoring assumed inexpensive

**CMMI**
- External customer
- Stable environment
- Critical system
- High cost of failure
- Validate and verify
- Fixed price
- Programming is an industrial process
- Documented plans, quantitative control
- Explicit knowledge
- Refactoring assumed expensive

None of these assumptions are as black and white as they appear.
Framework Assumptions and Focuses

- Agile
  - How
  - Project
  - Learning in projects
  - Short-term view

- CMMI
  - What
  - Organization
  - Learning at many levels
  - Long-term view
Fighting Words

CMMI / Plan-driven Methods
- Managed, controlled process
- Linear, sequential process
- Replicable, universal process
- Deterministic and goal-driven process

Agile / Light Methods
- Random, opportunistic process
- Overlapping, simultaneous and gaps
- Completely unique
- Negotiated and capricious

Why can’t we all just get along!
Complete, Align, Integrate

Use existing processes (where appropriate)

- SDLC, Agile, Scrum
- Internally developed
- CobiT, ItIL, PMI

Synthesize and hybridize based on organizational needs

Consistent, repeatable, Processes with measured results across all work

www.davidconsultinggroup.com
©2013 David Consulting Group
Alistair Cockburn’s “Barely Sufficient” Philosophy

• Is this something we really must do? If so, then what is the simplest thing we can do to satisfy this request.
  – The Agile / Waterfall Cooperative
# Combining Agile and CMMI for Project Planning

<table>
<thead>
<tr>
<th>CMMI Practice Coverage</th>
<th>Agile Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>PP1.1 WBS</td>
<td>Product &amp; Sprint Backlogs,</td>
</tr>
<tr>
<td>PP1.2 Estimate attributes</td>
<td>Function Points, SNAP, Planning poker</td>
</tr>
<tr>
<td>PP1.3 Lifecycle</td>
<td>Sprints, Planning, Retrospectives</td>
</tr>
<tr>
<td>PP1.4 Estimate effort/cost</td>
<td>Function Points, Planning poker</td>
</tr>
<tr>
<td>PP2.1 Budget/schedule</td>
<td>Release plan, Sprint plan, Velocity</td>
</tr>
<tr>
<td>PP2.2 Identify risks</td>
<td>Daily standup, Product Backlog</td>
</tr>
<tr>
<td>PP2.3 Data Management</td>
<td></td>
</tr>
<tr>
<td>PP2.4 Neccessary resources</td>
<td>Release plan, Sprint plan</td>
</tr>
<tr>
<td>PP2.5 Knowledge and skills</td>
<td>Sprint plan (weak with hybridization)</td>
</tr>
<tr>
<td>PP2.6 Stakeholders</td>
<td>Product Owner, Demo participants</td>
</tr>
<tr>
<td>PP2.7 Overall plan</td>
<td>Release plan, Sprint plan, Overall plan</td>
</tr>
<tr>
<td>PP3.1 Review plans</td>
<td>Planning game, Burn down/up</td>
</tr>
<tr>
<td>PP3.2 Reconcile the plan</td>
<td>Planning game, Velocity</td>
</tr>
<tr>
<td>PP3.3 Commitment</td>
<td>Planning game, Standup</td>
</tr>
</tbody>
</table>
## Example Two: Requirements Development

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Typical Artifact(s)</th>
<th>SCRUM Equivalent</th>
<th>Comments</th>
<th>Notes for PPQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Develop High Level Requirements</td>
<td>High-level requirements</td>
<td>Product Backlog</td>
<td>The Product Owner/Manager (or their designee) prepare the Product Backlog as a prioritized list of High-level requirements.</td>
<td>Ensure that the Product Backlog on the team site is prioritized and clear. Validate that requirements are traceable.</td>
</tr>
<tr>
<td>Initiate Requirements</td>
<td>Standard Requirements Document</td>
<td>Sprint Planning Meeting</td>
<td>The Product Manager shall contact the stakeholders shall determine the number of scrum teams and sprints needed to complete the project based on the items on the product backlog and their estimates.</td>
<td>Ensure the Product Backlog is maintained in priority order, that high level estimates are assigned and there is a record of the product manager/stakeholder discussion.</td>
</tr>
<tr>
<td>Develop Business Requirements</td>
<td>Standard Requirements Document</td>
<td>Product Backlog</td>
<td>The Product Owner/Manager includes Business Requirements in the Product Backlog; they may do this in the course of a Sprint Initiation meeting if they have not been previously identified. If more detailed requirements are required, a member of the Team should contact the product owner for clarification</td>
<td>Ensure that requirements are clearly communicated and traceable.</td>
</tr>
</tbody>
</table>
“If one has strong discipline without agility, the result is bureaucracy and stagnation. Agility without discipline is the unencumbered enthusiasm of a startup company before it has to turn to a profit.”
Websites:

Books:
- “Lean Thinking,” by James P. Womack and Daniel T. Jones
- “Lean Software Development,” by Mary and Tom Poppendieck
- “Agile Project Management with Scrum,” by Ken Schwaber
- “Agile Management,” by David J Anderson
- “Out of the Crisis,” by W. Edward Deming
Vocabulary, Part 1

- **Retrospective** – The sprint retrospective meeting is held at the end of every sprint after the sprint review meeting. The team and Scrum Master meet to discuss what went well and what to improve in the next sprint.

- **Demo** – The sprint demonstration is conducted after which the team asks the product owner whether (s)he feels the goals were met.

- **Velocity** – How much product backlog effort a team can handle in one sprint.

- **Burn Down** – Depicts the total task hours remaining per day. This shows you where your team stands regarding completing the tasks that comprise the product backlog items that achieve the goals of the sprint.

- **Burn Up** – A burn-up chart is a graphical representation that tracks progress over time by accumulating functionality as it is completed.

- **Product Owner** – A single person must have final authority representing the customer's interest in backlog prioritization and requirements questions.

- **Scrum Master** – Is a facilitator for the team and product owner. Rather than manage the team, the Scrum Master works to assist both the team and product owner.
Vocabulary, Part 2

- **Product Owner Proxy** – A stand in for the product owner.
- **Sprint cycle / Iteration** – An iteration of work during which an increment of product functionality is implemented.
- **Sprint team** – A team (or "Scrum team") is optimally comprised of seven (plus or minus two) people. For software development projects, the team members are usually a mix of software engineers, architects, programmers, analysts, QA experts, testers, UI designers, etc. This is often called "cross-functional project teams."
- **User Story** – A description of desired functionality told from the perspective of the user or customer (by **ROLE**).
- **Theme** – A collection of related user stories.
- **Epic** – A large user story (a full function).
Contact Us

Email: info@davidconsultinggroup.com

Phone: +1.610.644.2856

www.davidconsultinggroup.com

@DavidConsultGrp

/DavidConsultGrp

/company/David-Consulting-Group