

Agile Quality Management Techniques Complementing CMMI®

Presented by Jim Jamieson

9/25/2012

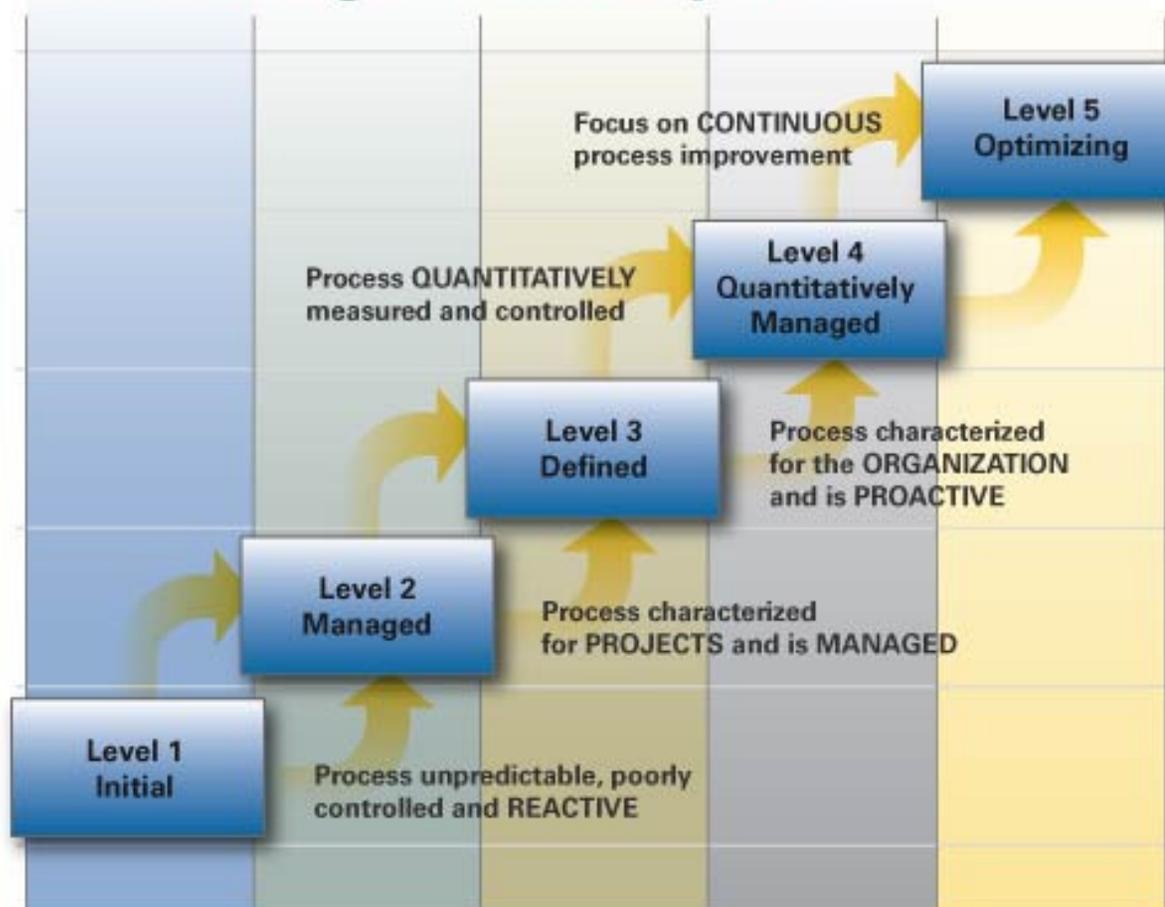


Background

- CMMI® aims to define and mature project processes by focusing on
 - Continuous quality and performance improvements
 - Deliver Quality Software
- Agile software development focuses on
 - Rapidly delivering high-quality software
 - That meets both the needs of the customer and
 - The goals of the organization over multiple iterations of the development lifecycle.
- The difference?
 - CMMI defines what you need to do, agile defines how you can do it faster

CMMI[®] Model

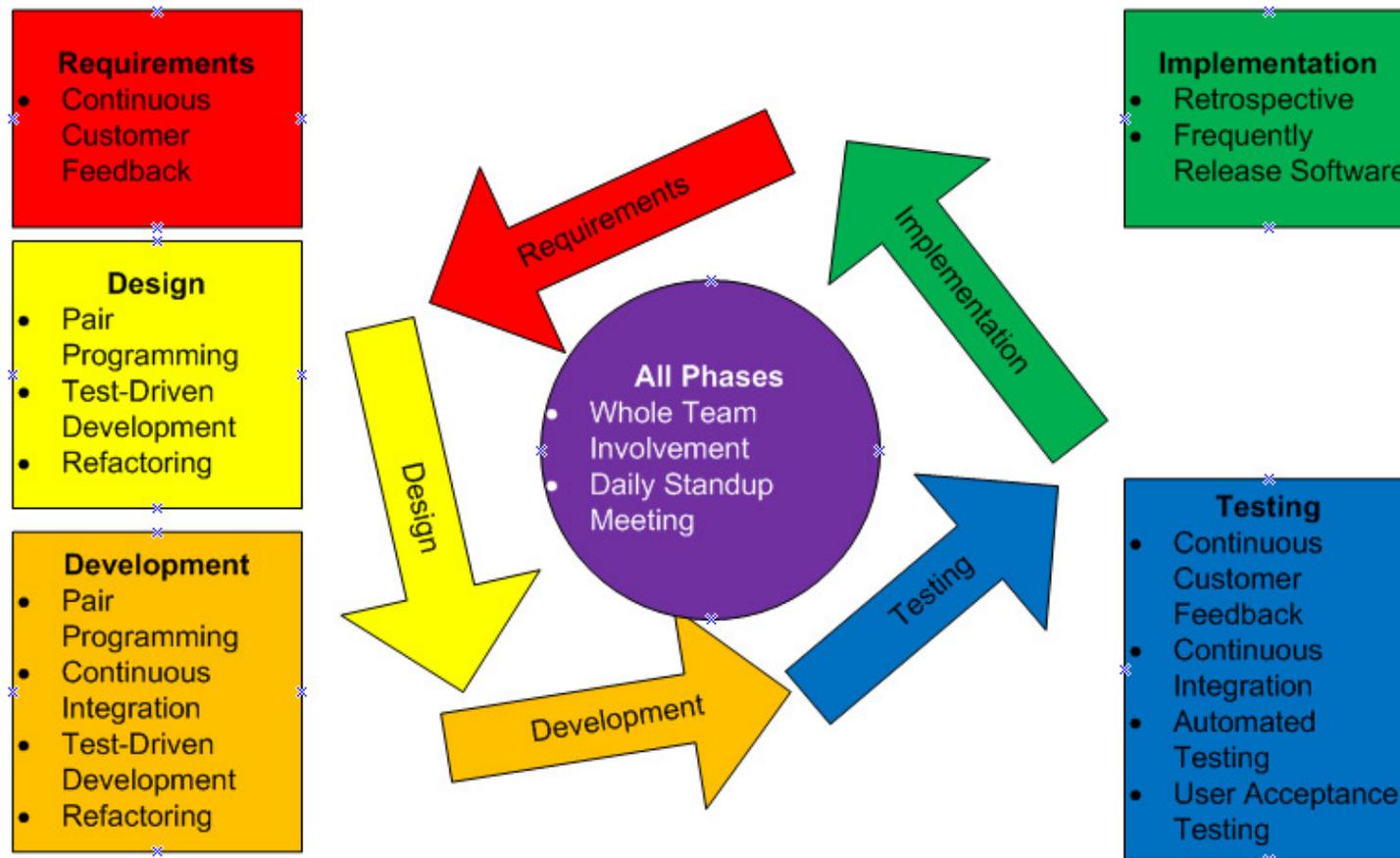
CMMI Staged Maturity Levels



Agile Techniques

- Whole team involvement
- Continuous customer feedback
- Pair programming
- Continuous integration
- Automated testing
- Test-driven development
- Refactoring
- User acceptance testing
- Retrospective
- Daily standup meeting
- Frequently release software

Agile Techniques in the SDLC





Whole Team Involvement

- Benefits
 - Better planning estimates
 - More detailed requirements
 - Gain commitment
- Potential Pitfalls
 - Increase in communication channels
 - Schedule delay
- Relationship to CMMI®
 - PMC, IPM, RD, REQM, PI, TS, VER, VAL
 - All phases of the SDLC result in increased involvement and commitment by the team to deliver the product



Continuous Customer Feedback

- Benefits
 - Willingness to discuss all project aspects openly
 - Quickly adapt application to suit customer's needs
- Potential Pitfalls
 - Key stakeholders may not wish to be involved
- Relationship to CMMI®
 - PMC, IPM, RD, VAL
 - By having the customer help to develop requirements they are better prepared to identify defects in the product



Pair Programming

- Benefits
 - Helps developers stay focused and think through things aloud
 - Produces higher quality code and reduces defects
 - Reduces maintainability
- Potential Pitfalls
 - Not everyone works together, should not be forced
- Relationship to CMMI®
 - TS, VER, VAL
 - Ensures best technical decisions are made and work can easily be checked to ensure it adheres to standards



Continuous Integration

- Benefits
 - Can be used to identify defects, integration errors, and failed tests earlier in the development process
 - Code is always “ready” for the customer
 - Provides continual feedback on the state of the application
- Potential Pitfalls
 - If everyone is not on board, then developers may be unwilling to fix something that is not their responsibility
- Relationship to CMMI®
 - CM, PI, VER
 - Frequent builds of the application ensure code is built the right way and all aspects integrated
 - Can be used to enforce standards



Automated Testing

- Benefits
 - Catches bugs early, when work is fresh
 - Safety net when refactoring
 - When integrated as a part of the build process, they can be run as a part of Continuous Integration
- Potential Pitfalls
 - Automated tests say nothing about the quality of the test
- Relationship to CMMI®
 - VAL: Code is continually validated to ensure it is correct



Test-Driven Development

- Benefits
 - Automated tests written first that will fail, code is written, tests run again to ensure code passes
 - Lower defect rate
 - Further requirement identification
- Potential Pitfalls
 - Can slow down development time
 - All developers must agree
- Relationship to CMMI®
 - VAL, TS: All solutions support automated tests to ensure the functionality is correct



Refactoring

- Benefits
 - Small transformations that constantly improve the code
 - Automated tests can support refactoring
 - Code does not degrade over time, easy to understand, maintain, and change
- Potential Pitfalls
 - Perceived lack of business value
- Relationship to CMMI®
 - PI/TS: Provides strategy for when to refactor by focusing on small transformations

User Acceptance Testing

- Benefits
 - Customers verify the specification they provided the developers has been met
 - Defects identified and fixed prior to release
- Potential Pitfalls
 - Customer schedule can drive when and how often UAT is performed
- Relationship to CMMI®
 - VAL: Customer validates product frequently to ensure business requirements are met



Retrospective

- Benefits
 - Improvements to project processes can continually be identified
 - Understanding where something is essential for improving quality
- Potential Pitfalls
 - Too many suggestions can actually hinder improvement
- Relationship to CMMI®
 - PP, PMC: Project is continually monitored to ensure the plans are met and an identified weakness will then lead to improvements for the next cycle



Daily Standup Meeting

- Benefits
 - Discussion of obstacles leads to resolutions
 - Everyone is on the same page with the project status
- Potential Pitfalls
 - Potential to become too detailed, must remain concise
- Relationship to CMMI®
 - IPM, PMC, RSKM: Project status including risks and issues faced by the team are discussed daily



Frequently Release Software

- Benefits
 - Customer can frequently evaluate changes to ensure they fit within business requirements
 - Identifies hidden business requirements
 - Increase business value and stakeholder confidence
- Potential Pitfalls
 - Customer rollout process could be long
 - Relies on many other techniques discussed previously
- Relationship to CMMI®
 - RD, TS, VER, VAL, PMC
 - Feeds back into next iteration to identify work to be done



Agile Impact on Software Development

- Automation as much as possible
- Reduction in schedule and cost
- Issues identified earlier in all processes
- Increased customer satisfaction
- Decrease defects
- Companies using these techniques have been certified as high as CMMI[®] Level 5

Where to Start?

- Start with techniques focused on communication
 - Whole team involvement
 - Continuous customer feedback
 - Retrospective
 - Daily standup meeting
- Move to some easy technical techniques
 - User acceptance testing
 - Frequently release software
 - Continuous integration
- Finally add advanced technical techniques
 - Pair programming
 - Automated testing
 - Refactoring
 - Test-driven development

Where to start?

- For existing projects a few recommendations
 - Continuous integration
 - Find ways to take build process and development standards to help team verify product is built the right way
 - Pair programming
 - Does not always have to be “programming” could simply be unit testing
 - Automated testing
 - Focused automated tests on difficult business requirements where there are a number of different scenarios and manual testing would be time consuming
 - Refactoring
 - Small/Incremental refactors instead of large “wholesale” changes as business requirements are added/modified



Summary

- Agile techniques can coexist with CMMI®
- Agile techniques can help reduce time to market while delivering higher-quality software
- While the techniques do focus on quality and a reduction in defects they also provide other benefits
 - Flexibility to meet customer needs
 - Maintainability of code
 - Shorter development time

Reference:

- Any questions?
- Jamieson, J.M. & Fallah, M.H. (March 2012). Agile Quality Management Techniques. *Software Quality Professional, Volume 12, Issue 2.*
- <http://www.cmmilevels.com/>