

Configuration Management and the Principles and Values of Agile Software Development

Stephen Vance
Stellar Advances
October 12, 2004

Agile Software Development

- Agile Methods
 - Low overhead
 - Low ceremony
 - Examples:
 - Extreme Programming
 - Scrum
 - DSDM

Extreme Programming (XP)

- Earliest
- Well-documented
- Common philosophy with other Agile methods as captured in the Agile Manifesto

Software Configuration Management

- Agile emphasizes “individuals and interactions over processes and tools”
 - May suggest SCM unimportant
 - XP emphasizes low-technology methods
- Well-chosen SCM practices support and enhance Agile methods

XP's Four Variables

- Variables
 - Cost
 - Time
 - Quality
 - Scope
- Risks affect variables; SCM helps address risks

XP's Four Values

- Values
 - Communication
 - Simplicity
 - Feedback
 - Courage
- SCM supports values; values guide SCM implementation

CM and the Principles and Values of Agile Development

Communication Supported

- SCM tool is medium for communication through code supporting continuous integration and collective ownership
- SCM support for continuous integration
 - Workspace management
 - ClearCase dynamic views
- Unit tests encapsulate requirements and design
- Integration with IDE improves efficiency

Communication Applied

- Effective use of SCM
- Minimize documentation
- Directory structures
 - Captured in SCM repository
- Branching policies
 - SCM tool can capture explanations
 - Perform branch specs
 - ClearCase config spec comments

Communication Applied (cont'd)

- Builds denoted through
 - Labels
 - Changelist numbers
 - Date/time queries
 - Release branches
- Task branches support sharing of larger stories

Simplicity Supported

- SCM tool saves overhead of manual processes
- SCM tool has its own overhead, but less than manual management
- IDE integration simplifies SCM usage
- Justified through
 - Versioning
 - Easy recovery of past releases
 - Change status tracking

Simplicity Applied

- Use SCM patterns
- Only apply patterns that are immediately necessary or reasonably anticipated
 - SCM refactoring harder than code refactoring suggests slightly more anticipation
- Use SCM tool facilities rather than external documentation to capture intent
- More complex SCM patterns help Agile scale

Feedback

- Unit tests are a major component of developer feedback; SCM tool supports as the repository for this code
- Builds, triggers and monitoring daemons enforce XP's coding standards practice
 - Style checking
 - Continuous integration test execution
 - Testing coverage analysis
- Automate reporting on change and development status
- Integration with work tracking updates story completion

CM and the Principles and Values of Agile Development

Courage

- SCM tool is the safety net
- Branches can support prototyping multiple alternatives without interfering with other work
- “Throwaway code” can be recoverable
- Refactoring can be strongly supported
 - Perforce’s Inter-File Branching™ allows a single file to branch to multiple targets

Conclusion

- SCM tools and processes can support Agile methods
- Agile principles apply to create and streamline SCM implementations
- Apply Agile principles to create SCM implementations that support Agile methods

Resources

- Manifesto for Agile Software Development, <http://www.agilemanifesto.org>
- Beck, Kent, Extreme Programming Explained: Embrace Change, Addison-Wesley, 2000
- Vance, Stephen, “Advanced SCM Branching Strategies,” http://www.vance.com/steve/perforce/Branching_Strategies.html