Software Test & Performance CONFERENCE

October 19 - 23, 2009

SCM for Agile Teams

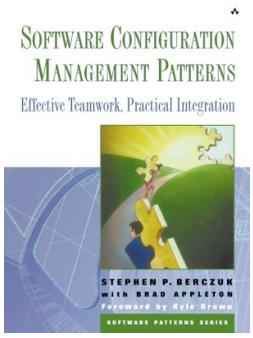
Steve Berczuk, Engineer, Humedica

Software Test & Performance CONFERENCE

Steve Berczuk

- www.berczuk.com/blog
- www.scmpatterns.com
- steve@berczuk.com

CSP ScrumAlliance CERTIFIED ScrumPractitioner



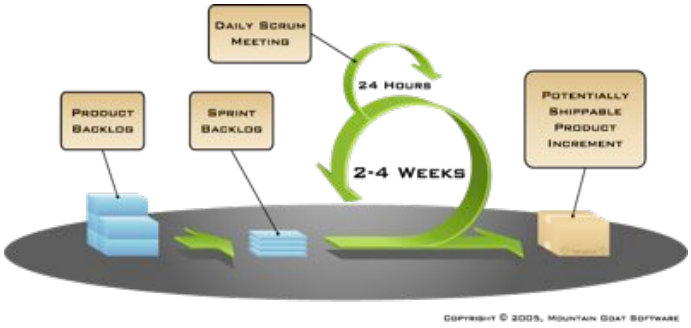
Agenda

- Agile/SCM
  - How they relate.
  - Why we care
- SCM Concepts & Common Problems
- Key Patterns and Practices
- Example

What is Agile?

- Incremental and Iterative
- Self-Organizing Teams
- Commitment
- Feedback & Adapting to Change
- Focus on delivering value

Scrum Process



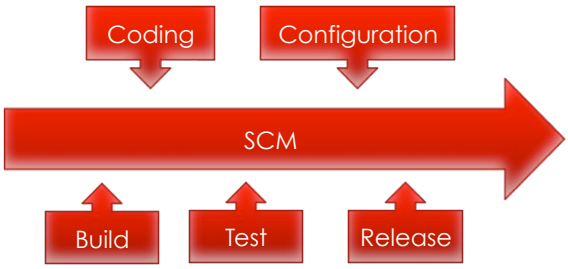
SCM

- What is SCM?
- Why do you care?
- Challenges
- What's different for Agile Teams?

What is SCM?

- *Functions:*
  - *Identification*
  - *Control*
  - *Status Accounting*
  - *Audit and Review*
- Activities:
  - Version Management
  - Branching and Merging
- **Assembly**
- **Change Management**

Communications Bus



SCM in the Lifecycle

- Setting up a workspace
- Creating a component (Integration Build)
- Validating against expected functionality
- Deploy
- Managing Parallel Releases/Maintenance
- Providing support information
- **Challenge: Right Amount of Process**

Not Enough Process

- "Builds for me!"
- "Works for me…!"
- "Another build is broken!"
- Ad-hoc code sharing
- Inconsistent releases.

Process in the Way

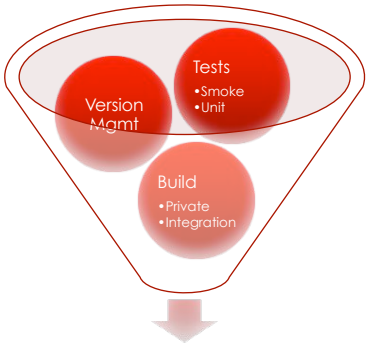
- Infrequent, Large commits
- Code Freeze
- Stabilization
- Long "Integration"

Agile Manifesto (and SCM)

- **Individuals and Interactions** *over* Processes and Tools
- **Working software** *over* Comprehensive Documentation
- **Customer Collaboration** *over* Contract Negotiation
- **Responding to Change** *over* Following a Plan

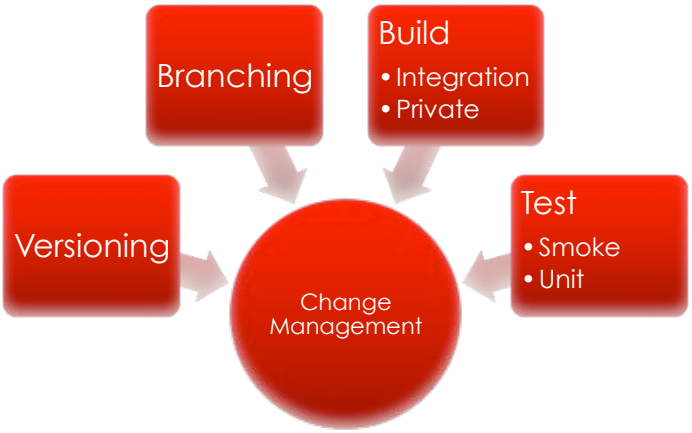
Good SCM is essential for Agile Software Development

Small Changes, Scales

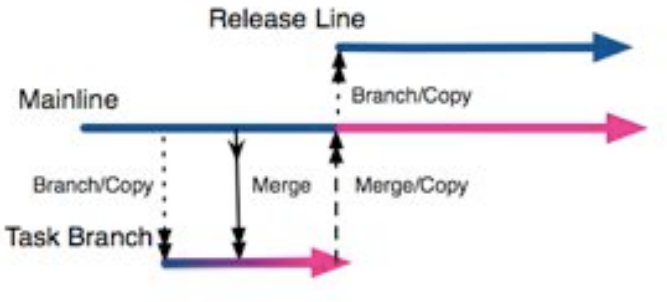


Application

Change Management



Codelines & Change



Parallel Work

Branch

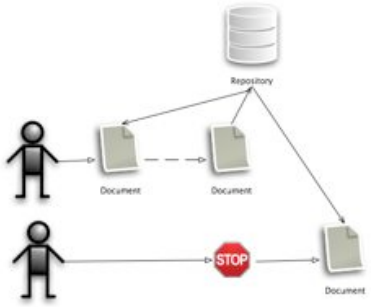
- Copy
- A start of parallel work

Merge

- Integrate
- Selective
- Override
- During Work
- At End

Locking Models

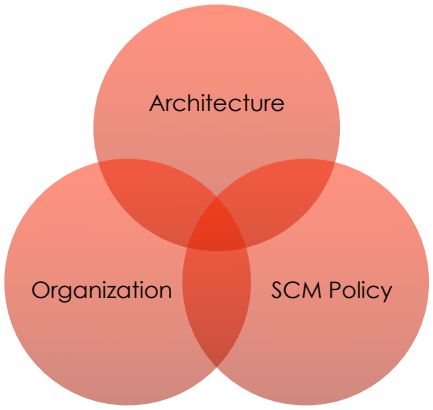
- Pessimistic Locking
  - Lock, Modify, Unlock
  - One editor at a time
  - Need to override locks



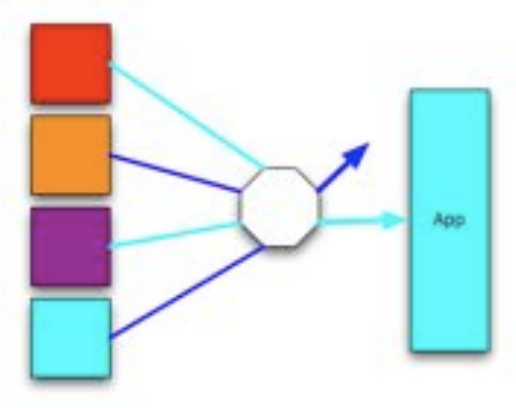
Locking Models

- Optimistic Locking
  - Copy, Modify, Merge
  - Concurrent Development

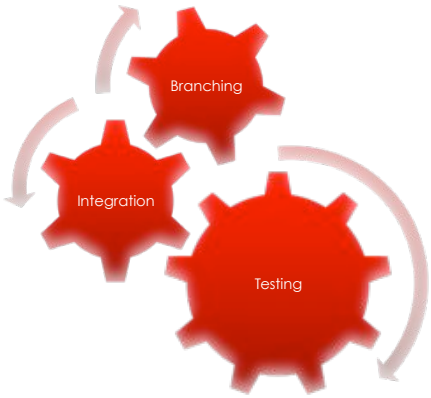
SCM In Context



Modular Configuration



SCM Practices



Tools

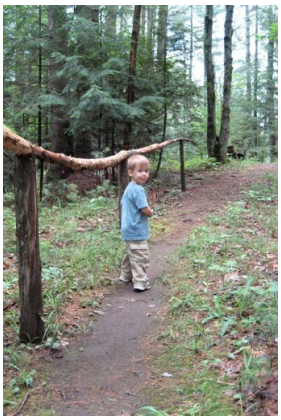
- Tools Help
- Process/Workflow First
- Qualities
  - Fast
  - IDE Integration
  - Issue Tracking Integration
  - Build Integration
  - Automation

Key Development Practices

- Codeline Structure
- Private Workspace
- Build
- Test
- Deploy
- (Automate)

Small Steps

- Changes
- Integrations
- Commits
- Modules
- Teams



Active Development Line

- One Development Stream
- Balance Stability, Progress
- Integrate!

Repository & Workspace

- Reproducibility
  - Development Workspace
  - Integration Workspace
- Repository:
  - VCS
  - Maven
- Roles:
  - Change Management
  - Tracing

Integration Build

- Standard Environment
- Automated
- Continuous
- Testing
- Needs to Work!
- Private Builds:
  - Similar Mechanism

Testing

- Unit Testing
- Integration Testing
- Regression Testing
- Test Configuration
- Balance Cost and Benefits

Essential Practices

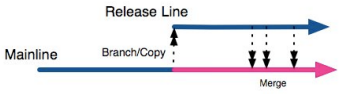
- Workspace Creation
- Build
- Continuous Integration
- Simple Codelines
- Tests
- Deployment

Example

- Finish a release without holding up work on the next release.

Release Prep Line

- Branch Early.
- Release 1 work on Branch. Use strict policy for commits.
- New Product work on MainLine.
- Relevant changes get "merged" from Branch to trunk as they occur.



Release Prep Codeline

Advantages

- Next Release on a stable codeline.
- New work on Mainline.

Disadvantages

- Merging is more difficult as time to release increases.
  - Mainline may change dramatically.
- Merge adds to overhead for everyone.
  - Can increase time to release.
- Team has divided focus.

Mainline (Agile)

- Branch late.
  - Don't branch until release is done.
- Keep team focused on release.
- Isolate features by components and configuration.

Mainline (Agile)

Advantages

- No branching until last possible moment.
- Team focused on release.
- No merging to add drag.

Disadvantages

- Requires good testing discipline to meet goals.
- Requires design and refactoring skill to isolate features by component.

Task Branch

- Branch Late.
  - Keep release on Mainline.
- New work onto task branch
- Those working on branch responsible for merges from Mainline.
- At release,
  - Create release branch from Main
  - "Copy" task branch to Mainline

Task Branch

Advantages

- Fewer people affected by merge issues.
- Merge issues don't delay release work.
- Highest priority is on Mainline

Disadvantages

- Mainline must be backed by tests to stay stable.
- Need an extra branch.

Decisions?

- Release Prep Branch
- Task Branch?
- Mainline?

Review

- Small, Repeatable Steps
- Deploy Early
- Branching is but one tool
- Test Frequently, and at all levels helps

Thank you!

Software Test & Performance CONFERENCE

For more information:

Steve Berczuk
Engineer

steve@berczuk.com
www.berczuk.com
www.berczuk.com/blog
www.scmpatterns.com