



Agile and Secure: Can We Be Both?

Dan Cornell, OWASP San Antonio **Leader**
Principal, Denim Group Ltd.
dan@denimgroup.com
(210) 572-4400

**OWASP
AppSec
Seattle**

Oct 2006

Copyright © 2006 - The OWASP Foundation
Permission is granted to copy, distribute and/or modify this document under the
terms of the Creative Commons Attribution-ShareAlike 2.5 License. To view this
license, visit <http://creativecommons.org/licenses/by-sa/2.5/>

The OWASP Foundation

<http://www.owasp.org/>

The Agile Practitioner's Dilemma

Agile Forces:

- More responsive to business concerns
- Increasing the frequency of stable releases
- Decreasing the time it takes to deploy new features



Secure Forces:

- More aggressive regulatory environment
- Increasing focus on need for security
- Traditional approaches are top-down, document centric



Objectives

- Background
- Goals of Agile Methods
- Goals of Secure Development Lifecycle (SDL)
- Review the Momentum of Agile Methods
- Look at An Integrated Process
- Challenges & Compromises



Background

- Programmer by background (MCSD, Java 2 Certified Programmer)
- Denim Group
 - ▶ Software development
 - ▶ Software security: vulnerability assessments, training mentoring
- Challenges facing our own agile teams
 - ▶ Deliver projects in an economically-responsible manner
 - ▶ Uphold security goals



Notable Agile Methods

- eXtreme Programming (XP)
- Feature Driven Development (FDD)
- SCRUM
- MSF for Agile Software Development
- Agile Unified Process (AUP)
- Crystal Clear
- Dynamic Systems Development Method (DSDM)



Manifesto for Agile Software Development

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

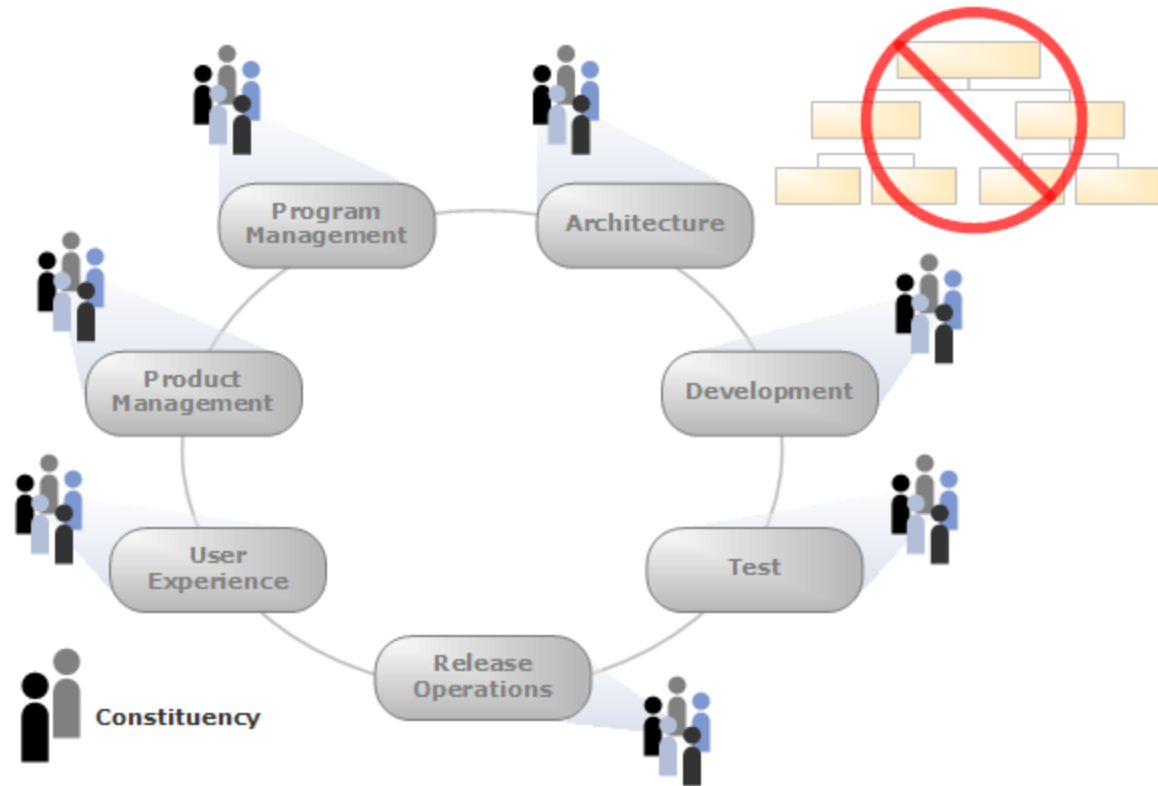
Responding to change over following a plan

Source: <http://www.agilemanifesto.org/>



Agile's Core Values

- Communication
- Simplicity
- Feedback
- Courage



Principles of Agile Development

- Rapid Feedback
- Simple Design
- Incremental Change
- Embracing Change
- Quality Work

- The system is appropriate for the intended audience.
- The code passes all the tests.
- The code communicates everything it needs to.
- The code has the smallest number of classes and methods.



Agile Practices

■ The Planning Game

- Customer: scope, priorities and release dates

- Developer: estimates, consequences and detailed scheduling

■ The Driving Metaphor

■ Shared Vision

■ On-Site Customer

- Development iterations or cycles that last 1-4 weeks.

■ Small Releases

- Release iterations as soon as possible (weekly, monthly, quarterly).



More Agile Practices

- Collective Ownership
- Test Driven
- Continuous Integration
- Coding Standards
- Pair Programming



Definition of Secure

A secure product is one that protects the confidentiality, integrity, and availability of the customers' information, and the integrity and availability of processing resources under control of the system's owner or administrator.

-- *Source: Writing Secure Code (Microsoft.com)*



A Secure Development Process...

- Strives To Be A Repeatable Process
- Requires Team Member Education
- Tracks Metrics and Maintains Accountability

Sources:

"Writing Secure Code" 2nd Ed., Howard & LeBlanc

*"The Trustworthy Computing Security Development Lifecycle"
by Lipner & Howard*



Secure Development Principles

- SD³: Secure by Design, Secure by Default, and in Deployment
- Learn From Mistakes
- Minimize Your Attack Surface
- Assume External Systems Are Insecure
- Plan On Failure
- Never Depend on Security Through Obscurity Alone
- Fix Security Issues Correctly



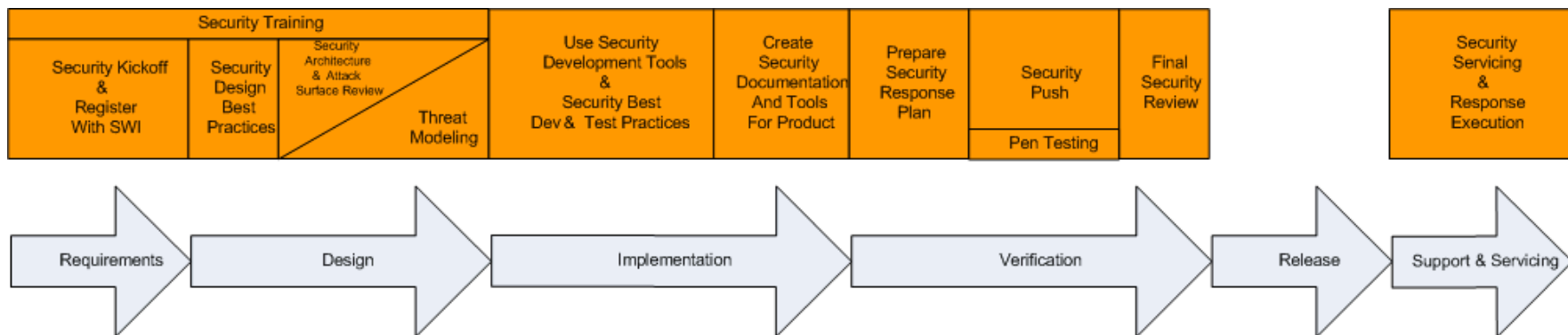
Secure Development Practices

- Education, Education, Education
- Threat Modeling
- Secure Coding Techniques
- Security Testing
- Security Code Reviews



Microsoft's Secure Development Lifecycle (SDL)

- Requirements
- Design
- Implementation
- Verification
- Release
- (Waterfall!)



Observations of the SDL in Practice

- Threat Modeling is the Highest-Priority Component
- Penetration Testing Alone is Not the Answer
- Tools Should be Complementary



Threat Modeling

■ STRIDE – classify threats

- ▶ Spoofing Identity
- ▶ Tampering with Data
- ▶ Repudiation
- ▶ Information Disclosure
- ▶ Denial of Service
- ▶ Elevation of Privilege

■ DREAD – rank vulnerabilities

- ▶ Damage Potential
- ▶ Reproducibility
- ▶ Exploitability
- ▶ Affected Users
- ▶ Discoverability



Dr. Dobb's says Agile Methods Are Catching On

41% of organizations have adopted an agile methodology

Of the 2,611 respondents doing agile...

- 37% using eXtreme Programming
- 19% using Feature Driven Development (FDD)
- 16% using SCRUM
- 7% using MSF for Agile Software Development

Source: <http://www.ddj.com/dept/architect/191800169>



Agile Teams are “Quality Infected”

- 60% reported increased productivity
- 66% reported improved quality
- 58% improved stakeholder satisfaction



Adoption Rate for Agile Practices

Of the respondents using an agile method...

- 36% have active customer participation
- 61% have adopted common coding guidelines
- 53% perform code regression testing
- 37% utilize pair programming



Let's Look at Some Specific Agile Methods

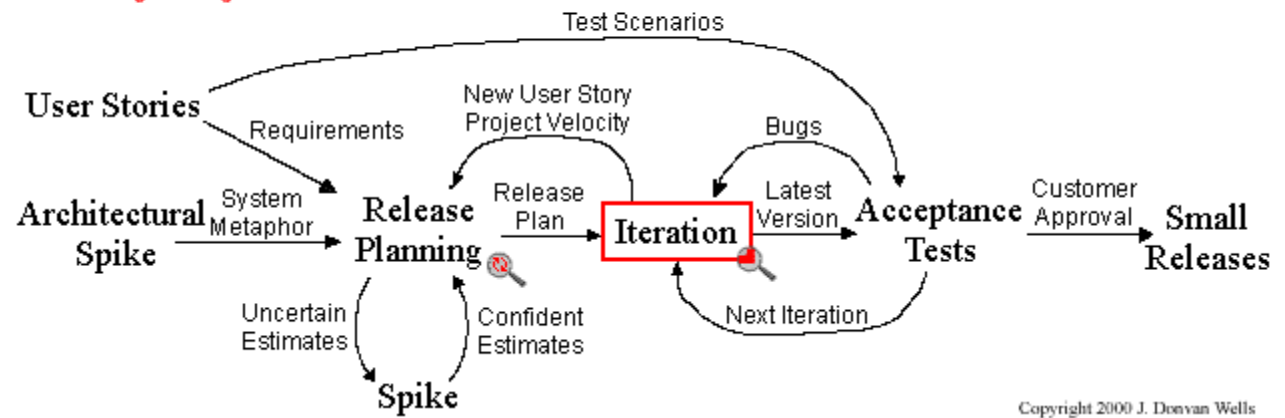
- eXtreme Programming (XP)
- Feature Driven Development (FDD)
- SCRUM
- MSF for Agile Software Development



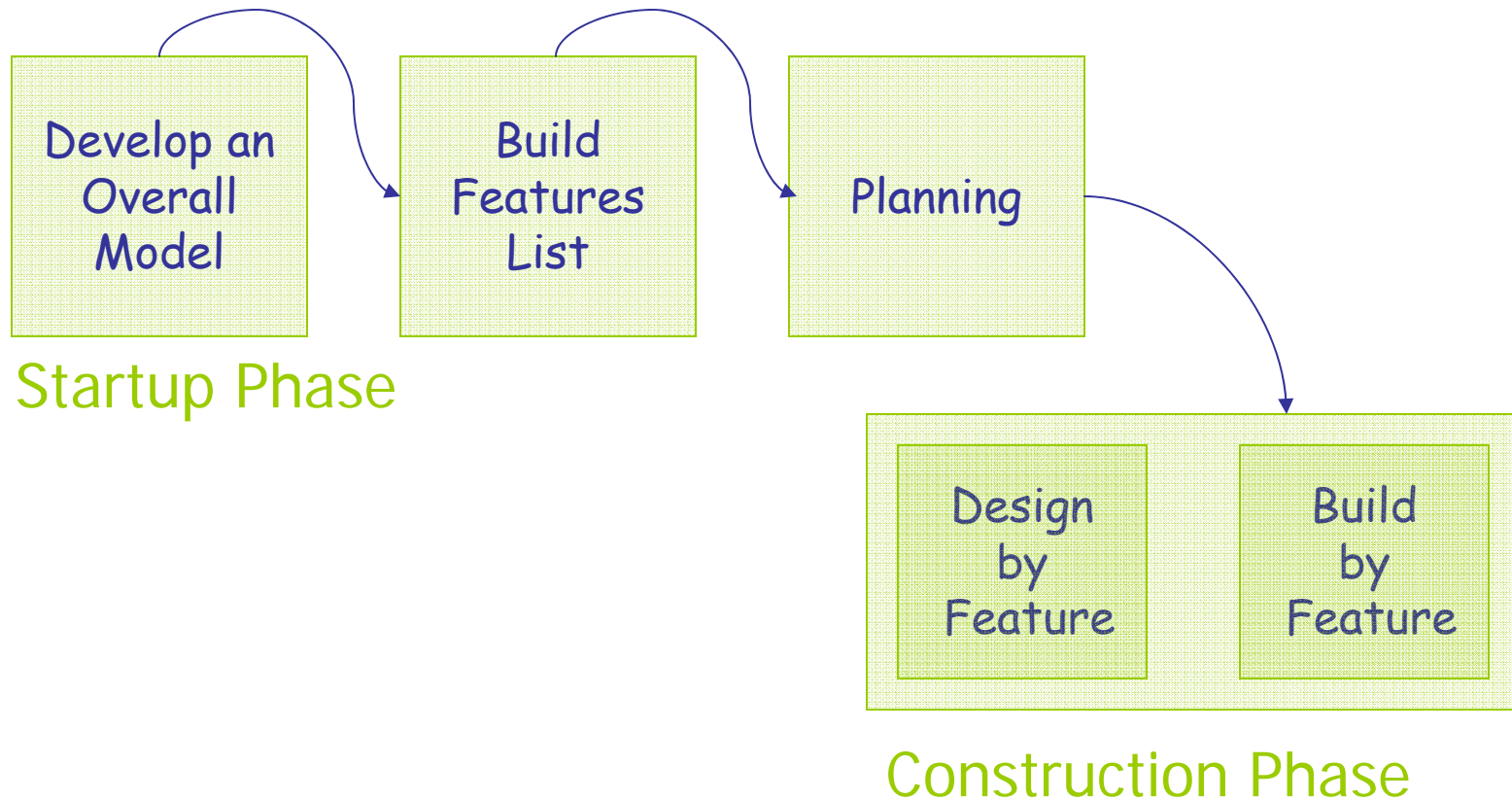
eXtreme Programming (XP)



Extreme Programming Project



Feature Driven Development (FDD)

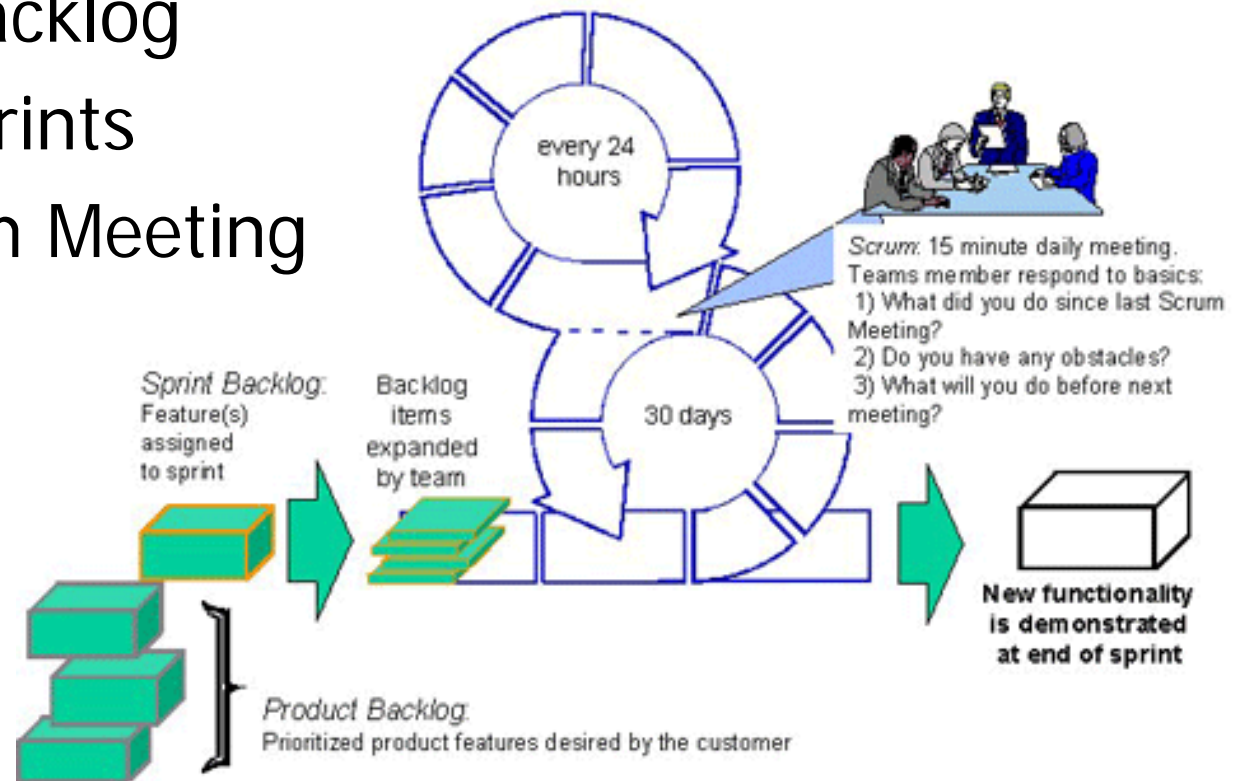


Source: <http://featuredrivendevelopment.com/>



SCRUM

- Commonly Used to Enhance Existing Systems
- Feature Backlog
- 30 Day Sprints
- Daily Team Meeting

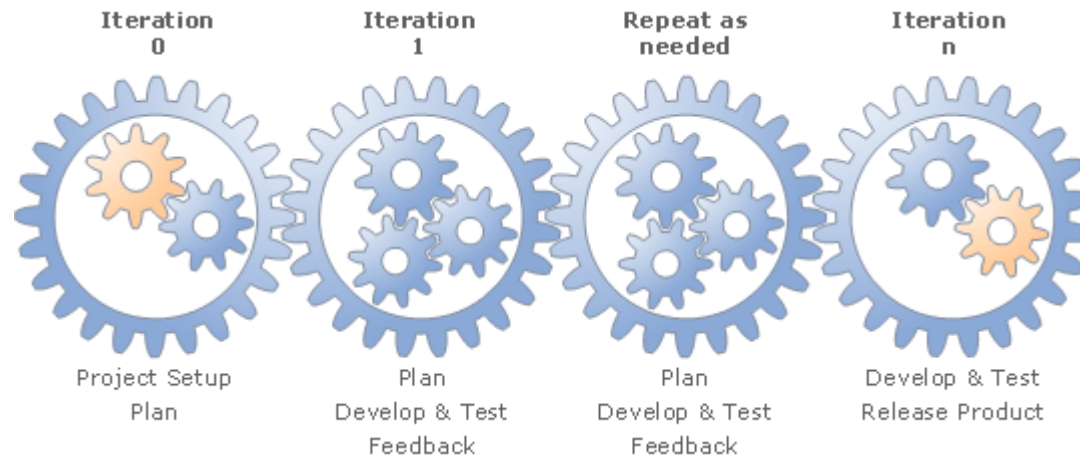


Source: <http://www.controlchaos.com/>

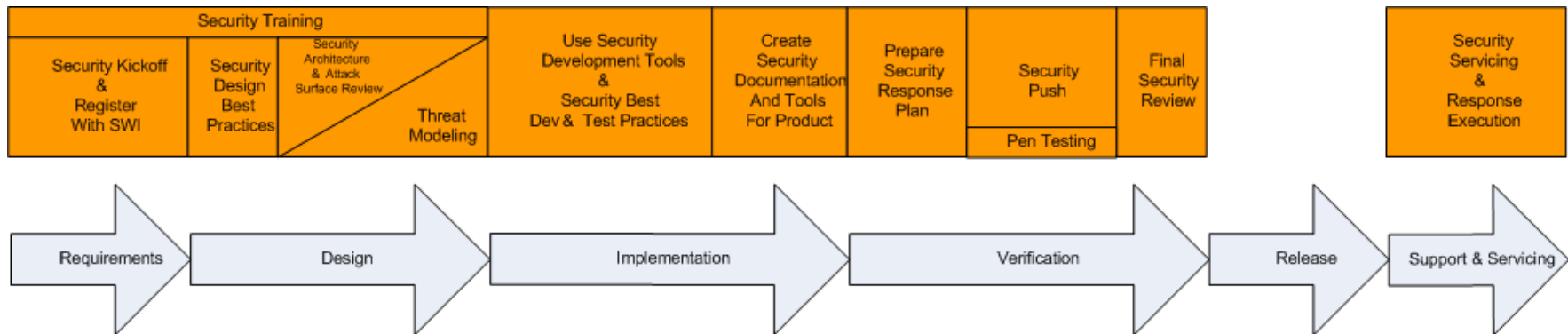


MSF for Agile Software Development

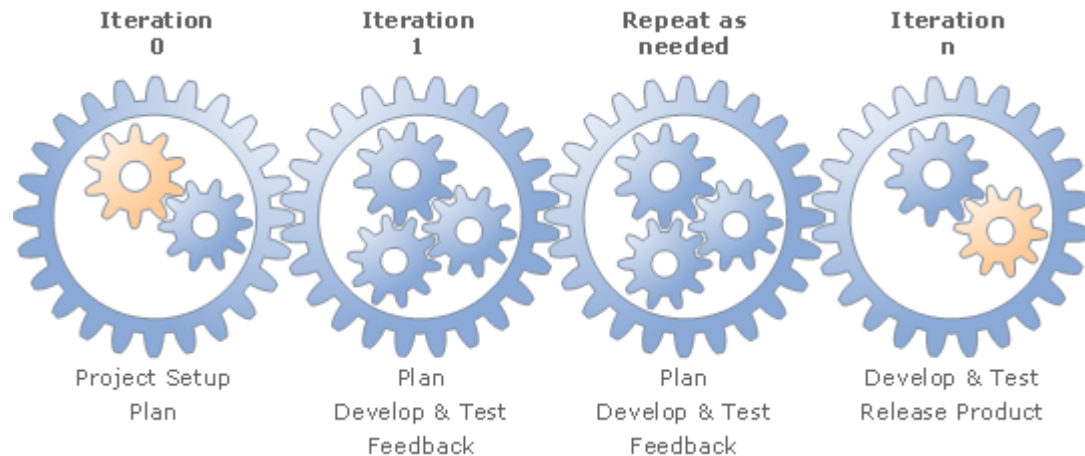
- Adapted from the Spiral / Waterfall Hybrid
- Product definition, development and testing occurs in overlapping iterations
- Different iterations have a different focus



An Integrated Process

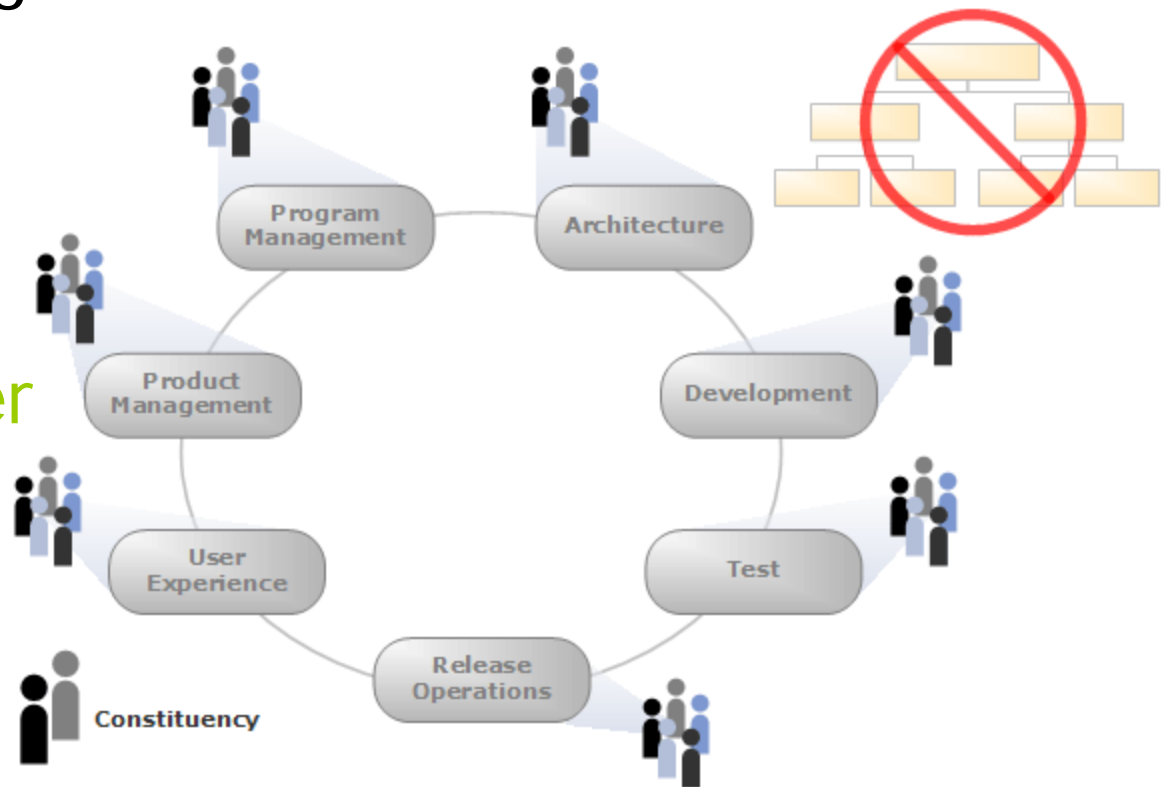


Making Agile Trustworthy



Project Roles

- Product Manager / Customer
- Program Manager / Coach
- Architect
- Developer
- Tester
- Security Adviser



Project Setup

- Education & Training *(include Security)*
 - ▶ Developers
 - ▶ Testers
 - ▶ Customers
- User Stories / Use Case Development
- Architecture Decisions (spikes)
- Agree on Threat Modeling standards for the project
 - ▶ STRIDE priorities
 - ▶ DREAD ratings



Release Planning

■ User Stories / Use Cases Drive...

- ▶ Acceptance Test Scenarios
- ▶ Estimations may affect priorities and thus the composition of the release
- ▶ Inputs for Threat Modeling
- ▶ Security Testing Scenarios
- ▶ Determine the qualitative “risk budget”
 - Keep the customer involved in making risk tradeoffs

■ Finalize Architecture & Development Guidelines

- ▶ Common Coding Standards (*include security*)
 - *Crucial for collective code ownership*
- ▶ Conduct Initial Threat Modeling (assets & threats)
- ▶ Designer’s Security Checklist



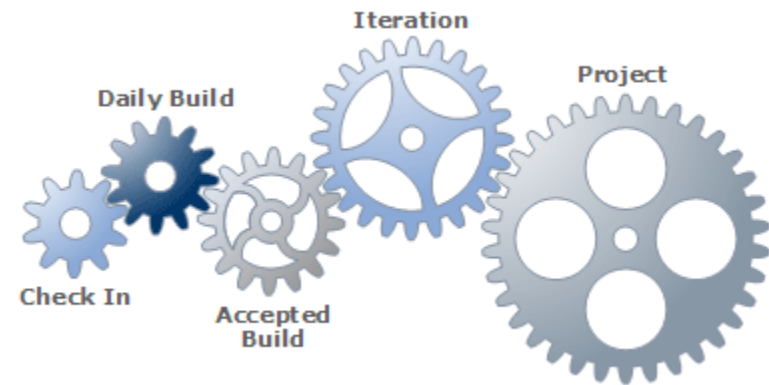
Iteration Planning

- 1-4 Weeks in Length (2 weeks is very common)
- Begins with an Iteration Planning Meeting
 - ▶ User Stories are broken down into Development Tasks
 - ▶ Developers estimate their own tasks
 - ▶ Document the Attack Surface (Story Level)
 - ▶ Model the threats alongside the user story documentation
 - Crucial in documentation-light processes
 - Capture these and keep them
 - Code will tell you what decision was made, threat models will tell you why decisions were made
 - Crucial for “refactoring” in the face of changing security priorities
- Never Slip the Date
 - ▶ Add or Remove Stories As Necessary



Executing an Iteration

- Daily Stand-ups
- Continuous Integration
 - ▶ Code Scanning Tools
 - ▶ Security Testing Tools



- Adherence to Common Coding Standards and Security Guidelines
 - ▶ Crucial for communal code ownership
- Developer's Checklist



Closing an Iteration

- Automation of Customer Acceptance Tests
 - ▶ Include negative testing for identified threats
- Security Code Review
 - ▶ Some may have happened informally during pair programming



Stabilizing a Release

- Schedule Defects & Vulnerabilities
 - ▶ Prioritize vulnerabilities with client input based on agreed-upon STRIDE and DREAD standards
- Security Push
 - ▶ Include traditional penetration testing



Compromises We've Made

- Feature-focus in iterations removes some “top down” control
- More documentation than is required in pure Agile development
 - ▶ Security coding standards
 - ▶ Project-specific STRIDE and DREAD standards
 - ▶ User story threat models



Values of an Agile and Secure Process

- Communication
- Simplicity
- Feedback
- Courage
- Trustworthy



Questions

Dan Cornell

dan@denimgroup.com

Website: www.denimgroup.com

Blog: www.agileandsecure.com

