



build | integrate | secure

Grow with Denim Group



Agile and Secure – Can We Be Both?

InnoTech Austin

October 11th, 2007

Agenda

- Background
- Evolution of Software Development Methodologies
- Benefits of Agile Development
- Requirements for Secure Development
- Agile and Secure
- Questions

Background

- Programmer by background
 - *Both .NET and JEE: MCSD and Java 2 Certified Programmer*
 - *Developer focused on security, not a security professional looking at development*
- Denim Group
 - *Software Development: .NET and JEE*
 - *Software / Application Security*
 - Vulnerability Assessments, Penetration Testing, Training, Mentoring
- Basis for this presentation:
 - *Work with our customers doing SDLC security and mentoring*
 - *Challenges facing our own development teams*
 - Deliver projects in an economically-responsible manner
 - Uphold security goals

Evolution of Software Development Methodologies

- Ad Hoc
- Waterfall
- Agile

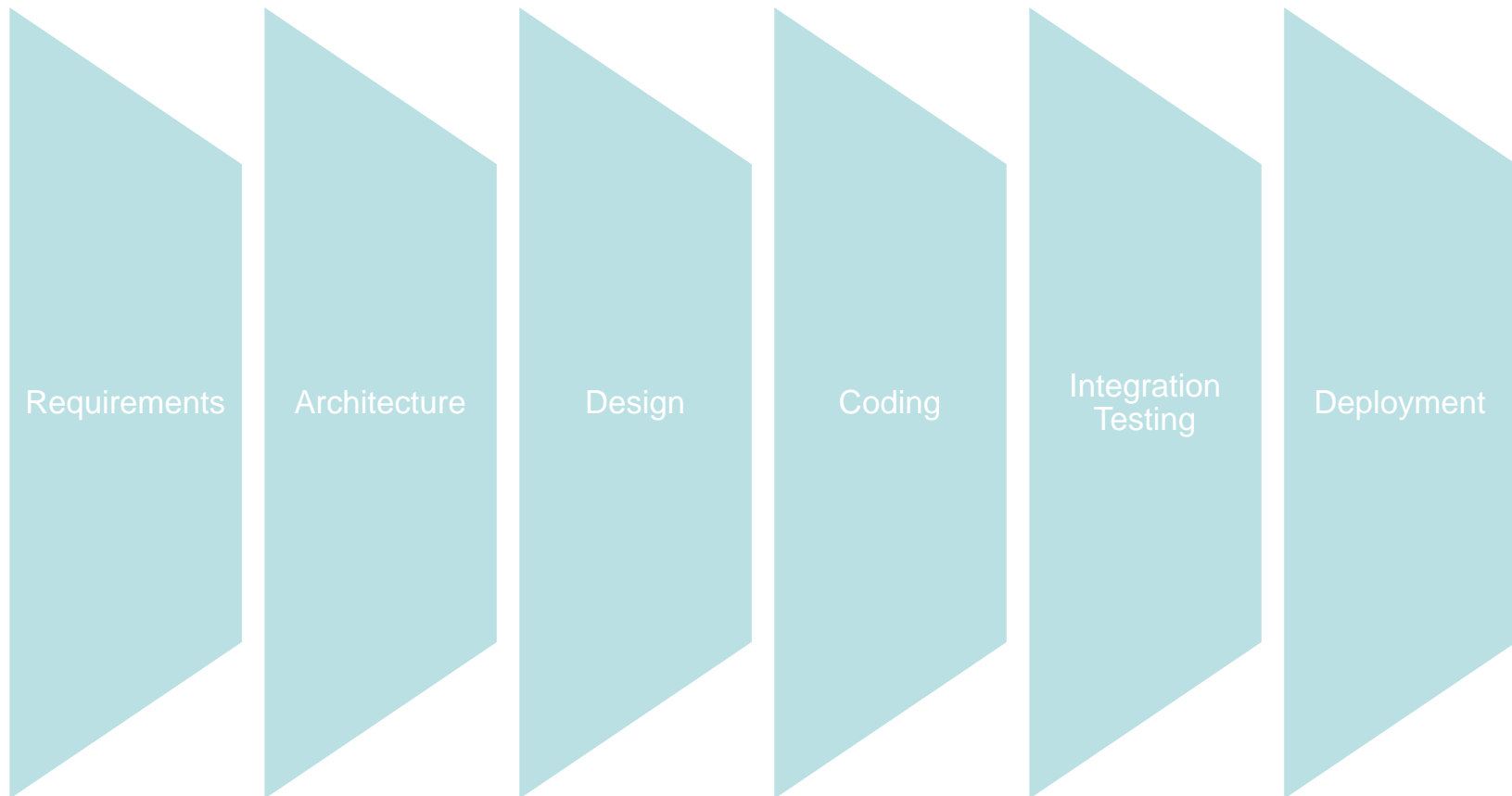
Ad Hoc Software Development

- Early days of computing
 - *Focus was on hardware*
 - *Software was secondary*
- No structure – “cowboy coding”
- Became unacceptable as software systems became larger and more critical

Waterfall Software Development

- Treat software engineering as a structured engineering process
- House building metaphor

Waterfall Software Development



Problems with Waterfall Model

- Creating software is different than creating bridges or buildings
 - *Creativity required throughout the process – not just at the outset*
- Very documentation heavy
- Changes are expensive
 - *Must go back up the waterfall for impact analysis*
- Business requirements change over time
 - *By the time you finish a system, the target has moved*

Enter Agile Methods

- Be more responsive to business concerns
- Increase the frequency of stable releases
- Decrease the time it takes to deploy new features
- Do not waste time on “superfluous” documentation and planning

Notable Agile Methods

- eXtreme Programming (XP)
- Feature Driven Development (FDD)
- SCRUM
- MSF for Agile Software Development
- Agile Unified Process (AUP)
- Crystal

Manifesto for Agile Software Development

Individuals and interactions over processes and tools

Working software over comprehensive documentation

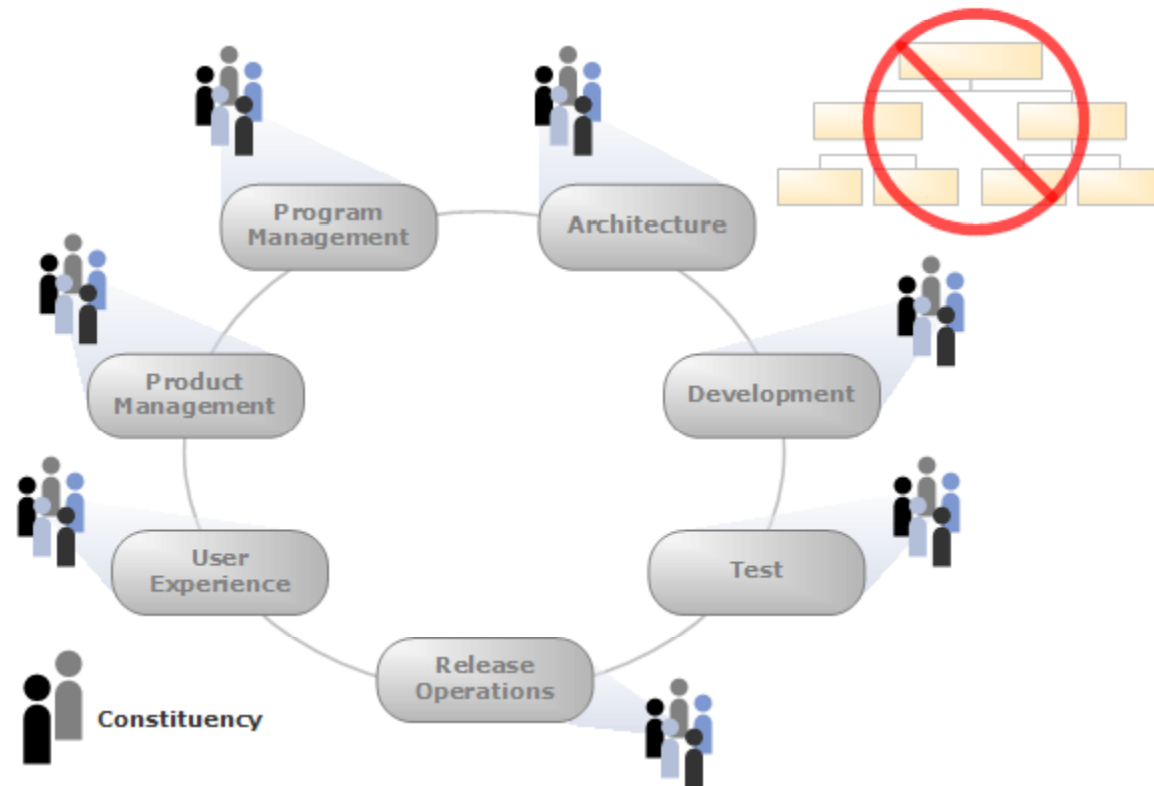
Customer collaboration over contract negotiation

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
- The Driving Metaphor
- Shared Vision
- On-Site Customer
- Small Releases

• Customer: scope, priorities and release dates

• Developer: estimates, consequences and detailed scheduling

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

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

More Agile Practices

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

The Agile Practitioner's Dilemma

Agile Forces:

- More responsive to business concerns
- Increasing the frequency of stable releases
- Decrease the time it takes to deploy new features
- Do not waste time on “superfluous” documentation and planning



Secure Forces:

- Comply with more aggressive regulatory environment
- Focus on need for security
- Traditional approaches to security require additional documentation and planning (D'Oh!)

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

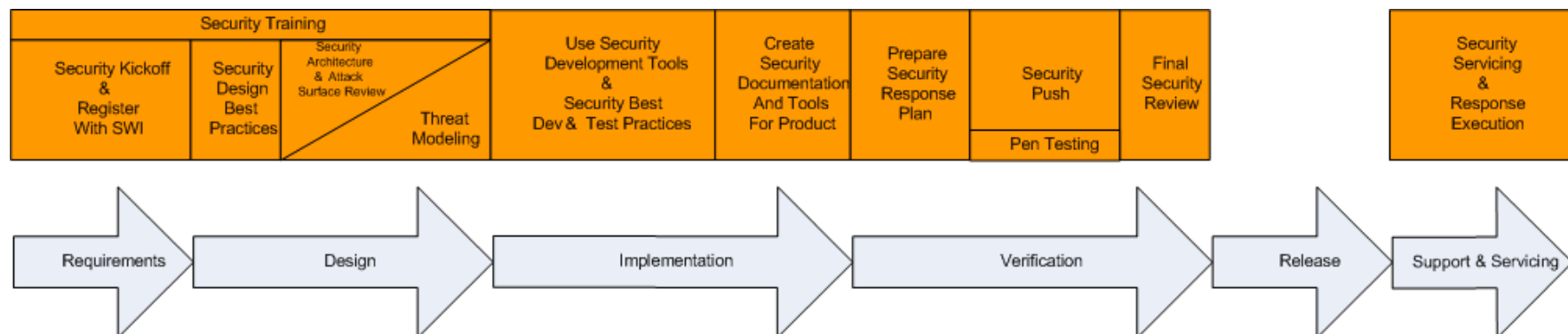
- SD³: Secure by Design, Secure by Default, and in Deployment
- Learn From Mistakes
- Minimize Your Attack Surface
- Assume External Systems Are Insecure
- Plan On Failure
- Never Depend on Security Through Obscurity Alone
- Fix Security Issues Correctly

Secure Development Practices

- Threat Modeling / Architectural Risk Assessment
- Education, Education, Education
- Secure Coding Techniques
 - *Via standards, practitioner knowledge and tools*
- Security Reviews
 - *Architecture*
 - *Design*
 - *Code*
- Security Testing (Penetration Testing)

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>

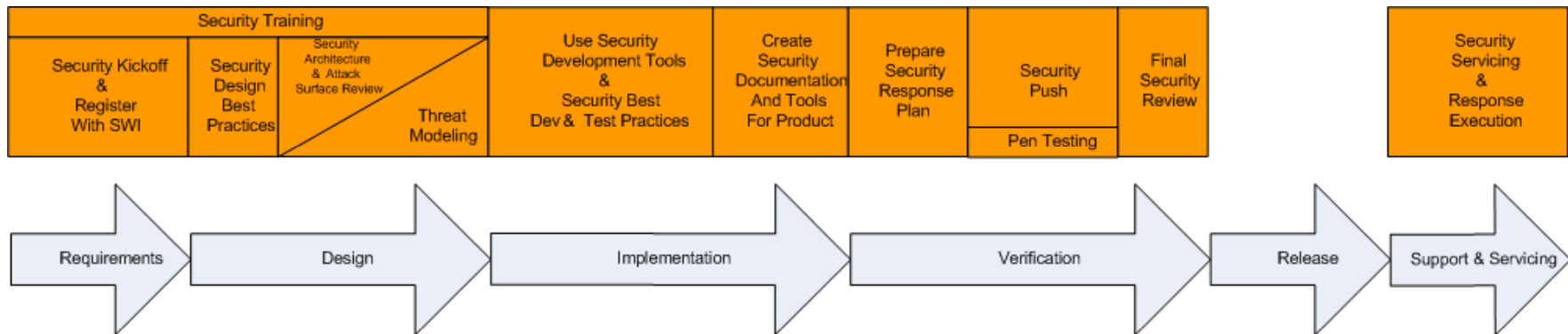
Agile Teams are “Quality Infected”

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

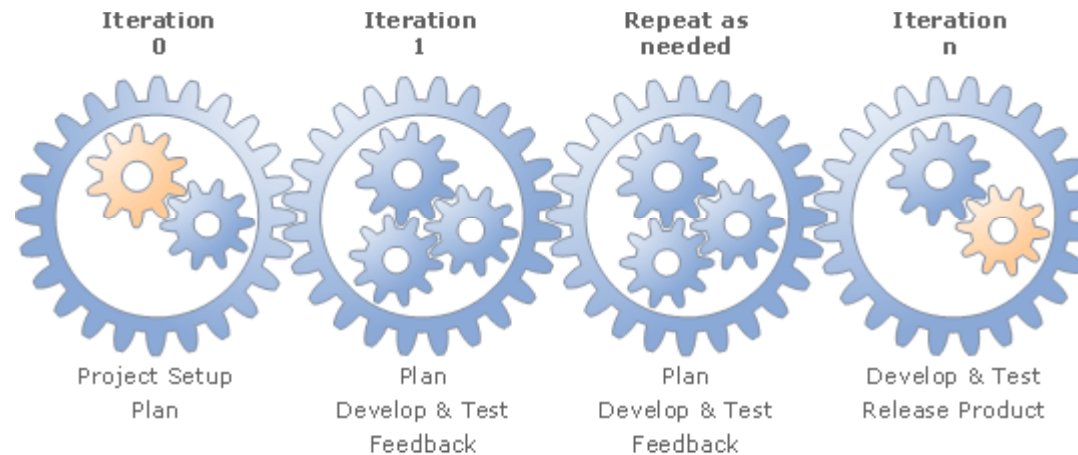
Adoption Rate for Agile Practices

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

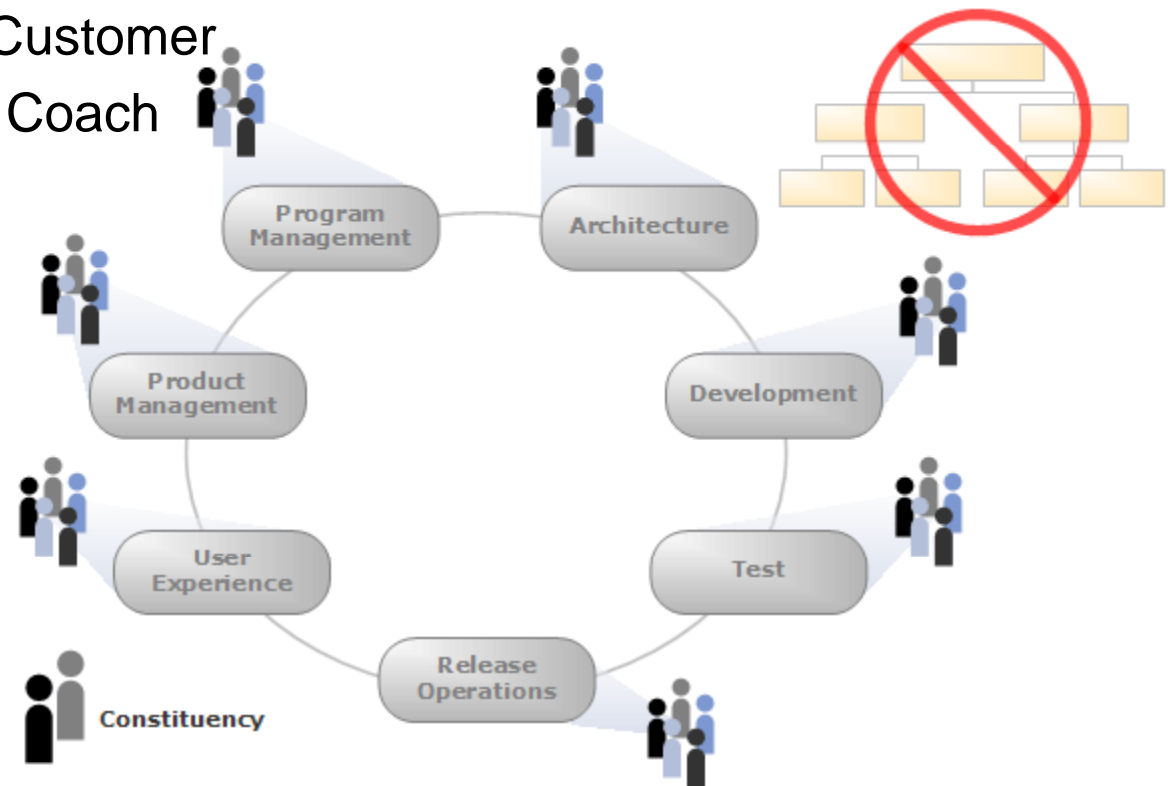


Making Agile Trustworthy



Project Roles

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



Organization Setup

- Education & Training (*include Security*)
 - *Developers*
 - *Testers*
 - *Customers*
- User Stories / Use Case Driven Process
- Enterprise Architecture Decisions
- Organizational Adoption of Threat Modeling

Project / Release Planning

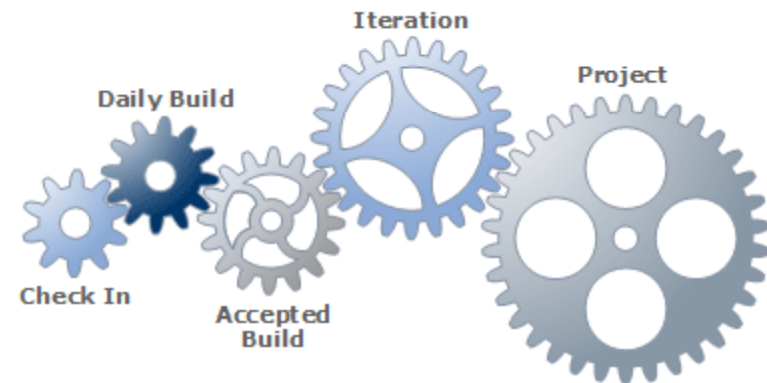
- User Stories / Use Cases Drive...
 - *Acceptance Test Scenarios*
 - *Estimations may affect priorities and thus the composition of the release*
 - *Inputs for Threat Modeling*
 - *Security Testing Scenarios*
 - *Determine the qualitative “risk budget”*
 - *Keep the customer involved in making risk tradeoffs*
- Finalize Architecture & Development Guidelines
 - *Common Coding Standards (include security)*
 - *Crucial for collective code ownership*
 - *Data Classification standards*
 - *Conduct Initial Threat Modeling (assets & threats)*
 - *Agree on STRIDE and DREAD classifications*
 - *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

- Security Compromises:
 - *Short term, iterative focus removes some “top down” control*
 - *Focus on individual features can blind the process to cross-feature security issues*
- Agile Compromises:
 - *More documentation than is required in pure Agile development*
 - Security coding standards
 - Data classification standards
 - Project-specific STRIDE and DREAD standards
 - User story threat models
 - *Additional tasks increase development time*
 - *Forces customers to accept security (isn't this a good thing?)*

Values of an Agile and Secure Process

- Communication
- Simplicity
- Feedback
- Courage
- Trustworthy

Questions

Dan Cornell

dan@denimgroup.com

(210) 572-4400

Website: www.denimgroup.com

Blog 1: www.agileandsecure.com

Blog 2: denimgroup.typepad.com