Consciously Evolving an Agile Team
Aaron Ruhnow
Technical Lead, ScrumMaster
iLevel, Structural Frame Software (SFS)
Aaron.Ruhnow@weyerhaeuser.com
What We’ll Discuss…

- The principles generated by one team of developers that allowed them to become self-directed and cross-functional
- How these changes assisted in engaging the rest of the project team.
- “Conscious Evolution” – braking out of traditional practices into the agile mindset.
- We WON’T discuss specific practices and processes we adopted except for a few examples.
Part of iLevel’s “Structural Frame Software” Group (SFS)

Project teams develop applications to interactively “size” iLevel products in residential structures

Traditionally used waterfall processes

Relationships sometimes adversarial
The Development Team/Project

- Team of 8 developers, two off-site
- “Personas” sometimes hindered the teams’ ability to collaborate and become cross-functional
- Able to “Start Small”
- Implemented the standard Scrum practices
Collaboration and sharing happens only occasionally

“Shared Resources” (SQA, Engineers, Architect, DBA) work with the team.
Communication between groups on a formal basis only. Adversarial relationship sometimes exist between silos.
Can We Be Agile?

How do we become self-directed and cross-functional?

How do we eliminate the silos?

How do we make better software and avoid the conflicts and mistakes of the past?
To Become Agile...

Change in Attitude/Mindset

- Principles
  - “Everyone is responsible for Quality”
  - Allow the team to determine how to get the work done
  - Collaboration over documentation
  - Learn from mistakes, don’t just correct them
  - Start Small – use “advance teams”
  - Shared conscience – no code reviews or “handing over the wall”

- Shared conscience – no code reviews or “handing over the wall”
To Become Agile…

- Break the silos
  - Pair Programming
  - Shared Workspace
  - Elimination of set roles
  - Amplified Learning
- As a Team, Establish Coding Qualities
  - TDD Development
  - Create “Testable” Code
  - Good OO design (patterns)
  - Etc…
“Conscious Evolution”
“Conscious Evolution”

- Get Incremental
  - Short iterations for opportunities to evolve
  - Test/Requirements/Code/Feedback loop
  - Improve code as you go on

- “Constructive Conflict”

- Clarity/Transparency

- Focus

- Evolve the Leadership
Emphasis on Quality

“Entire Team is Responsible for Quality”

Put quality on the same level as scope and schedule

Move quality activities from the end of the process to the front (get “Test Driven”)

Quality efforts on same backlog as new features
Adversaries to Allies

- Product Owner part of Team
- “Embed” the shared resources
  - System Architect
  - Engineers
  - SQA Lead
  - DBA
  - Install Developer
Product Owner as Part of Team

- Gradually establish a “common language” between developers and product owner
  - Use Cases evolved into User Stories
  - Requirements evolved into User Acceptance Tests
- Fully engaged in day-to-day workings
  - Daily standup
  - Frequent (daily+) review of progress
Feedback Loop/Blur the Cycle

- **Short iterations:**
  - Requires the team to change the way they developed software
  - Feedback from product owner and stakeholders is frequent and consistent
  - Provides a “rhythm” for the team
  - Many opportunities to fix what’s not working
    - especially in early iterations
  - Mistakes, unexpected issues and changes are smaller and more manageable

- **Continuous Integration (CI) critical**
EVOLUTION: Requirements
Requirements...the REALLY Old Days
Use Case – Input Floor Performance Information

ID: VUC-560

Goal (Task) Name: Input Floor Performance Information

Primary Actor: Beam User

Supporting (Secondary) Actors / Stakeholders: None

Description/Context: The user has the ability to view and adjust the Floor Performance (TJ-Pro and CVC) modifiers (i.e. strapping, ceiling, perpendicular partitions, bridging, extra flange, etc). The user has the ability to adjust the minimum TJ-Pro Rating number. The user has the ability to choose which type of decking sheathing to use for a particular range of on center spacings.

Scope Level: Sub-System Black-box

Scope: New Beam Problem Input

Goal (Task) Level: User

Pre-conditions:
- User has a design problem opened.
- Program defaults, design defaults, and problem defaults are already defined and system is set to floor.
- User is designing a residential or commercial floor joist system using a member with a parallel profile.

Minimal Guarantees: No input options will prevent the user from attempting to design the member (i.e. the system must ensure that any floor performance options chosen will not cause the design to fail in
3.1. Spans Input

3.1.1. Span Entry Form

FR1. The input fields and entered values for spans will be displayed in a vertically oriented table, with each column representing a different span. Here is an example of this input table:

As of 5/1/2006, span support justifications all on one line:

<table>
<thead>
<tr>
<th>Number</th>
<th>Left OH</th>
<th>Span 1</th>
<th>Span 2</th>
<th>Span 3</th>
<th>Right OH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td></td>
<td>24</td>
<td>322</td>
<td>333</td>
<td>12</td>
</tr>
<tr>
<td>Span Just.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As of 4/7/2006, span support justifications on separate lines:

<table>
<thead>
<tr>
<th>Number</th>
<th>Left OH</th>
<th>1</th>
<th>2</th>
<th>Right OH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td></td>
<td>10'</td>
<td>8'</td>
<td></td>
</tr>
<tr>
<td>Start Point</td>
<td></td>
<td>Center</td>
<td>Right</td>
<td></td>
</tr>
<tr>
<td>End Point</td>
<td></td>
<td>Left</td>
<td>Right</td>
<td></td>
</tr>
</tbody>
</table>

FR2. There will always be one blank span column to the right of the last-defined span column to provide a direct entry area for additional spans. The entry of a valid span length in the current blank column will trigger a new blank column to appear.

FR3. Heads-down keyboard entry of span lengths will be fully supported; keyboard keys will be assigned for necessary navigation and add-support functionality to allow the user to enter span lengths and add subsequent supports without looking up from the keyboard. Entry should preferably be condensed to one area of the keyboard to support one-handed entry.

FR4. The span entry form window area shall be designed to allow the user to view a minimum of 4 spans and both end overhangs without the need to scroll. Horizontal scrolling to see additional spans (after the first four) will be allowed.

FR5. Span support justification drop-down selection values should be made smaller by the use of understandable, but small icons, representing left, center, right and...
The requirements documents look nice…but do they truly reflect the current state of the application?
Hey...didn’t we code this already?

3. Functional Requirements

3.1. Add a Hole

The following section will describe how a user can input a new hole.

FR1. The user will be able to navigate to a section of the application which will allow entry of holes into the member to be designed (design problem).

3.1.1. <Heading 3>

<Body Text>

• <List Bullet>

• <List>

• <List Bullet 2>

• <List 2>

○ <List Bullet 3>

○ <List 3>

FR1. <Reqmnt Number>

3.1.1.1. <Heading 4>

<Body Text>

1. <List Number>

1. <List>
StoryPage167

Story: VS-167
Use Case: UC-548, FA:
Story Text: As a stakeholder holes are only allowed in "joist" member types and TJI® products. If rectangular product is selected: design the rectangular product, ignore the hole and note it in the report, update graphic to rectangular product and do not draw the hole. save hole information and re-populate if TJI is selected.

UA Test Cases:

- When member type is anything other than Joist, all holes are removed from grid and graphic, and elements within Hole Tab are disabled.
- When member type is Joist, and product selected is Rectangular, all holes are removed from graphic (all holes remain in grid and are enabled). ASSUMPTION: holes can still be sent to PAE, but will be ignored, and PAE will return message verifying hole was ignored, and Vitruvius will display)
- "Note: Holes are analyzed for parallel TJI series joists used in joist applications only." should be displayed on the Hole tab
- If rectangular product as joist member type containing holes is changed back to a TJI product, the holes are again visible (in graphic and grid) and sized by the PAE.
- If holes are entered to a joist member type, and then member type is set to a flush beam or drop beam, holes are lost and NO "Note: Holes are analyzed for parallel TJI series joists used in joist applications only." will be shown on the report.
**EVOLUTION...Tests and Requirements Together!**

**Start Vitruvius**

---

**Requirement** - When the member type is set to Joist, holes are visible and active. When the member type is changed to a Drop Beam, the holes are removed and not visible in the grid and graphic, and the controls in the hole tab are disabled.

---

**Included page:** [FitVitruvius.IncludedLibraryFiles.SetMemberTypeToJoist](#)

---

**Included page:** [FitVitruvius.IncludedLibraryFiles.SetAllHoleTypes](#)

<table>
<thead>
<tr>
<th>Holes Grid</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Button Click</td>
<td>Hole Number</td>
</tr>
<tr>
<td>Add Hole 1</td>
<td>1</td>
</tr>
<tr>
<td>Add Hole 2</td>
<td>2</td>
</tr>
</tbody>
</table>

---

**Included page:** [FitVitruvius.IncludedLibraryFiles.SetMemberTypeToDropBeam](#)

---

**Included page:** [FitVitruvius.IncludedLibraryFiles.CheckHoleTabControlsDisabled](#)

<table>
<thead>
<tr>
<th>Holes Tab</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Check Hole Count</td>
<td>0</td>
</tr>
<tr>
<td>Check Hole Grid Enabled</td>
<td>False</td>
</tr>
<tr>
<td>Check Hole Add Button Enabled</td>
<td>False</td>
</tr>
<tr>
<td>Check Hole Delete Button Enabled</td>
<td>False</td>
</tr>
<tr>
<td>Check Hole Copy Button Enabled</td>
<td>False</td>
</tr>
</tbody>
</table>
Conclusions

- Start Small
- Measured and evolutionary changes – not all at once
- Enable team to establish the qualities/principles/processes
- Foster communication with all team members/shared resources
- Continuous Integration to speed up feedback loops
- Don’t implement too many changes at once.
- “It’s not a destination but a journey”
Questions?