UX Engineering Process: Bill Scott

(Video: 0_Introduction.mp4): **Introduction**

- 00:00:00-00:03:11: Bill discusses what he looks for in a front-end engineer and lays out the schedule for the course.
    - Presentation: http://billwscott.com/share/presentations/2012/fews/ws.pdf

Part 1: Building Products (00:03:12-01:22:21)

(Video: 1_A.mp4): **UX Process**

- 00:03:12-00:04:40: Audience is polled to gauge their background and experience.
- 00:04:41-00:14:02: A few different audience members describe their company's UX approach and process.

(Video: 1_B.mp4): **Bill Scott and PayPal**

- 00:14:03-00:18:41: All developers have their own experiences and bring a unique approach toward UX and front-end engineering. Bill describes his journey from his development work in the 1980's all the way to today. Along the way he learned how to add efficiencies into teams of all sizes.
- 00:18:42-00:22:17: What can we learn from PayPal? Difficulties arise when a company has a culture of a long shelf life.

(Video: 1_C.mp4): **Break Down the Walls**

- 00:22:17-00:24:06: In a typical organization, walls exist between the product/business team, design team, and engineering team. The product team consists of product managers and business analysts.
- 0:24:07-00:29:35: The design team is typically includes UED, UX, ID, IA, VizDe, content and designer talent. Engineering teams are often front-end and user interface engineers along with web developers.
- 00:29:36-00:32:14: After deadlines are met, these teams typically disband and form new teams on new projects.

(Video: 1_D.mp4): **Netflix: Rapid Experimentation**

- 00:32:15-00:39:01: A large portion of this course discusses "lean UX". A lean UX will break down the walls between teams and allow for a more collaborative approach. At Netflix, Bill had a few people possible on each team. The culture was to higher smart, get products out the door as fast as possible and don't overthink it.

- 00:39:02-00:43:22: At Netflix, 90% or more of UI code was thrown away each year. It doesn't take much A/B testing to result in a lot of thrown away code. Inuit learned a similar lesson.

(Video: 1_E.mp4): **Lessons Learned**

- 00:43:23-00:47:26: Design for volatility. Planning for change can help with code reuse. Developers have to be able to build, test, and learn.
- 00:47:27-00:54:41: Software is always tearing itself apart. Recognize that different layers change at different velocities. Start with experience versus components.
- 00:54:42-00:55:12: Build in rapid experimentation. The UI layer is an experimentation layer. Early rapid prototyping leads to learning.

(Video: 1_F.mp4): **Q & A**

- 00:55:13-01:22:21: Q & A discussion around the topics discussed in Part 1. Questions asked by the audience include:
    - What are your thoughts on prototyping with pen and paper versus HTML?
    - At Netflix, how did you do usability testing?
    - What are your thoughts on documentation?
    - How do you deal with stakeholders' fear of change?
    - What advice would you give to someone trying to introduce A/B testing?

Part 2: Lean UX (01:22:22-02:50:53)

(Video: 2_A.mp4): **Lean Manufacturing**

- 01:22:22-01:27:42: Lean UX stems from the manufacturing revolution in Japan. It's the ideal of shrinking batch sizes and the acceleration of cycle times.
- 01:27:43-01:32:37: The book, The Lean Startup, looks at developing experimentation systems that allow teams to move at the speed of these systems instead of the speed of "Caesar giving a thumb's up or down".
- 01:32:38-01:36:05: Being lean means defining your minimum viable product and building it in small batches.

(Video: 2_B.mp4): **Lean UX**

- 01:36:06-01:40:43: Lean UX helps address the issues with the current UX process. Focusing on deliverables instead of experience results in waste. Even in agile development environments, design is often omitted from the process.
- 01:40:44-01:47:02: A look at lean UX at PayPal.

(Video: 2_C.mp4): **Three Key Principles for Lean UX**

- 01:47:03-01:52:29: Shared Understanding – The more understanding the less documentation.

- 01:52:30-01:53:51: Deep Collaboration – Having a strong belief that ideas come from many different voices.
- 01:53:52-01:58:47: Continuous Customer Feedback – This is the lifeblood of the team because it elimitates politics and turns a team outside-in.

(Video: 2_D.mp4): **A Healthy Product Lifecycle**

- 01:58:48-02:09:40: Members from each team participate in all phases of the product life cycle starting from discovery and finishing with delivery. This process can still be agile, but unlike a pure agile approach, lean UX focuses on a build/test/learn cycle.

(Video: 2_E.mp4): **Key Lessons from a Lean UX Team**

- 02:09:41-02:12:10: Lean is not necessarily agile. Agile focuses on engineering delivery while lean focuses on learning. See the workshops PDF for more information on agile vs. lean.
    - http://billwscott.com/share/presentations/2012/fews/wss.pdf
- 02:12:11-02:17:39: Co-locate if at all possible. Deeper collaboration and time with the customer increases feedback. Github adds a slight counterpoint but proves cooperation without coordination can work
    - http://tomayko.com/writings/adopt-an-open-source-process-constraints/
- 02:17:40-02:22:58: Create a team working agreement and sprint faster so you can deliver to customers as often as possible. Also, sketch to code.
- 02:22:59-02:24:50: Make the spec real. There are many prototyping tools available to create a living spec.

(Video: 2_F.mp4): **Example: Spotify**

- 02:24:51-02:28:50: Spotify has adopted many of the lean UX ideas. Spotify uses squads which are similar to a scrum team, but feel more like a lean startup. Tribes are a collection of squads that work in a related area. Chapters and guilds represent horizontal practices within or across tribes.
    - The Lean UX Advocate: http://www.jeffgothelf.com/blog
    - Lean UX article: http://uxdesign.smashing.com/2011/03/07/lean-ux-getting-out-of-the-deliverables-business/
    - Principle of Shared Understanding: http://52weeksofux.com/post/2403607066/building-a-shared-understanding
- 02:28:51-02:35:57: Q&A and wrap-up discussion about Part 2
    - Bill shares some stories about usability testing.

(Video: 2_G.mp4): **Q&A and wrap-up discussion, continued**

- 02:35:58-02:50:53: Additional questions and discussions about the lean UX techniques discussed in Part 2
    - What's the best way to validate or quantify customer feedback?

    - What brought Netflix to the prominent position it owns today?

Part 3: Lean Tech Stack (02:50:54-04:23:49)

(Video: 3_A.mp4): **The Way It Was**

- 02:50:54-02:58:06: Bill begins with a brief overview of his journey through software development. He was building games in 1985. Developing a UI was very difficult.
- 02:58:07-03:00:43: From 1985 through 2005 Bill did a lot of proprietary development. Then he moved into the open source world with frameworks like the Yahoo! Design patterns library and work with the YUI team.

(Video 3_B.mp4): **Using Open Source at PayPal**

- 03:00:44-03:07:20: PayPal is using technologies like Twitter Bootstrap and Backbone.js. Using an internal Github revolutionized their internal platform. Every developer is encouraged to experiment and generate repos.

(Video 3_C.mp4): **Portable UI Frameworks**

- 03:07:21-03:13:47: A portable UI can be delivered continuously and run on either the client or server. Their lean UI stack is a combination of Node.js (for prototyping) and Sparta 2.3 (for production).

(Video 3_D.mp4): **Implementing the Portable UI**

- 03:13:48-03:22:46: Implementing a portable UI across two different systems is the tricky part. Bill continues his PayPal example and describes how the two stacks were unified.
- 03:22:47-03:27:00: A lean stack must be independent of the backend language and flexible enough to run on either the client or the server. It should be good at building websites as well as web applications.

(Video 3_E.mp4): **Rapid Prototyping**

- 03:27:01-03:33:09: The first thing to remember is prototypes are not one-size-fits-all. Paper prototypes or a white board might be the right answer. However HTML is also great for prototyping. Using frameworks like Twitter Bootstrap maintain the right fidelity and create a fast, good-looking UI.
    - List of prototyping tools: http://bit.ly/SfWygk
- 03:33:10-03:38:33: Use Google Chrome for prototyping. Ask your users to do the same. Also, Node.js is great for application prototyping even if you aren't using it for production.

(Video 3_F.mp4): **Rapid Prototyping, continued**

- 03:38:34-03:47:16: JS templating because it makes all of your UI bits be based in JavaScript. This makes them portable to either the client or server and even deliverable by a CDN.

- 03:47:17-03:48:13: CSS can be a powerful tool for prototyping. Keep your CSS clean, though. Consider using something like LESS or Sass.
    - Gradient Generator: http://www.colorzilla.com/gradient-generator/
- 03:48:14-03:53:16: Bill suggests a number of other tools and techniques to make rapid prototyping easier and more effective.

(Video 3_G.mp4): **UI Architecture Concerns**

- 03:53:17-03:56:53: Which is better: client side or server side? It depends. For a page-to-page interaction, server side is typically better. Client side is best for an app or single-page design. Twitter is a good example of this argument.
    - http://engineering.twitter.com/2012/06/improving-performance-ontwittercom.html
- 03:56:54-04:02:40: The server-side or client-side argument also appears in discussions around application vs. page or native vs. web.

(Video 3_H.mp4): **Responsive Web Design vs. Responsive Server Side**

- 04:02:41-04:06:23: When creating an omni-channel experience, RWD is implemented with CSS media queries and made to adapt. The downside is the potential for too large a payload on devices.
- 04:06:24-04:10:34: Mobile first does not mean designing for mobile device in isolation. A mobile first approach is designing for an ecosystem of form factors through the prism of mobile.

(Video 3_I.mp4): **Q&A Discussion for Part 3**

- 04:10:35-04:23:49: Questions asked during the Part 3 Q&A session:
    - Is your mock data hard-coded or pulled dynamically from somewhere?
    - How can you take a responsive server-side mentality when rebuilding a desktop application?
    - For responsive web application, are you including anything like jQuery Mobile?

Part 4: Anti-Patterns (04:23:50-05:22:16)

(Video: 4_A.mp4): **Anti-Patterns 1-3**

- 04:23:50-04:26:21: Bill defines many different anti-patterns in lean UX and how to avoid them. You can avoid the *Genius Designer* by keeping their inspiration but focusing on your minimal viable product.
- 04:26:22-04:28:39: With small teams, many members work across disciplines. *Tribal Groups* form as the team grows and tribes form around each discipline.
- 04:28:40-04:32:01: It may be assumed that a *Newcomer* will automatically adapt but teams must be patient, answer questions and teach vocabulary to any new member. Recognize early if they are the right fit.

(Video: 4_B.mp4): **Anti-Patterns 4-7**

- 04:32:02-04:34:06: Team members can be *Addicted* to old/bad habits. New habits must be performed enough to ensure they are internalized.
- 04:34:07-04:37:07: The *Naysayer* can kill collaboration. If someone is not on board with the process, they may not be a good fit.
- 04:37:08-04:38:55: A *Visitor* can give crucial input during development. But watch out. A visitor can bring the same issues as a newcomer.
- 04:38:56-04:40:23: *Magic Tools* allow for easy prototyping and ideation. They can, however, cause people to work in isolation and ruin collaboration.

(Video: 4_C.mp4): **Anti-Patterns 8-11**

- 04:40:24-04:43:26: If a developer, product manager, or designer *Goes Dark* for more than a day or two, collaboration can be crushed. Limit isolation to short periods of time.
- 04:43:27-04:45:11: A *Change in Cadence* is good. However prepare your team for any rhythm changes to keep productivity loss to a minimum.
- 04:45:12-04:46:25: If work is not divided properly, you may end up with *Too Many Cooks* in the kitchen. Be sure to have clear decision makers.
- 04:46:26-04:51:06: *Not Enough Pizza* means the team is scaling faster that it can handle. Keep your team 2-pizza size.

(Video: 4_D.mp4): **Anti-Patterns 12-14**

- 04:51:07-04:53:52: Lean UX relies on shared understanding. *The Tower of Babel* arises when knowledge is assumed. Ask clarifying questions to make sure everyone is on the same page.
- 04:53:53-04:59:12: If *You've Got Mail*, your team has reverted to email for collaboration. Keep the collaboration high-bandwidth.
- 04:59:13-05:01:26: If engineers are driving design, then the *Inmates are Running the Asylum*. Make sure the front-end engineers are working with the product and design teams.

(Video: 4_E.mp4): **Anti-Patterns 15-18**

- 05:01:27-05:04:15: At trap easy for a designer to fall into is the *Perfectionist*. Iteration will help; so will a collaborative approach.
- 05:04:16-05:08:08: A team will stumble on the *Weakest Link*. Talent acquisition is the key and a team must of the freedom to replace talent.
- 05:08:09-05:09:06: *The Wall* between teams can happen. Smaller teams and collaboration keep the lines of communication open and the walls down.
- 05:09:07-05:10:28: A messy tech stack becomes T*angled-up Technology*. Keep the services and UI separate.

(Video: 4_F.mp4): **Final Q&A and Wrap-up**

- 05:10:29-05:22:16: Bill wraps up his presentation and fields questions from the audience.

- Steve Jobs was very combative in his approach. Can you contrast this mentality with lean UX?
- How do you handle version control? How would you like to handle version control?
- Do you have any tips for deployment?