

EECS 394: Overview of Final Report Answers

Each question was 8 points, for a total of 24. The median was 17.

Daily reports

Kudos to Matt Schuchhardt for the quote “if stories are so important to the project, they ought to be the ones speaking in the standup,” from <http://www.exampler.com/blog/2007/11/06/latour-3-anthrax-and-standups/>

Grading: How well did you pick up on clues to possible developer problems? Did you respond with probing questions or premature prescriptive instructions?

Done: Implemented dataTables jquery plugin to handle displaying, sorting and filtering tabular alumni data

Developer is focusing on technology, not on what users will be able to do

Doing: teleconference with client at noon.

Developer is confusing “other” work with user value

major coding marathon tonight

Potentially unsustainable binge programming

Doing: "As a designer, I can edit my profile name, location, profile picture, company, title, personal program reflection, personal quote, and email address."

OK. (Probably not “too big” since these look like simple form input fields. Saying “designer” for a design portfolio site is better than a generic “user.”)

Doing: Integrate the slideshow element into the design theme for improved Visitor experience.

(1) No clear user story. (2) “Improved” is untestable. How do you know you’re “done?”

Done: "public users can send email to students"

Fine, assuming “public user” is a client term.

Doing: still working on hooking up forms to fields in the database

(1) No clear user story. (2) “Still?” Are stories too big? Is some non-story taking away developer time?

Doing: An administrator can log in and add students or alumni to the system

Fine

Missing Expertise

Grading: Did you focus on team development, while not losing sight of the project and the business? Specific things I was looking for:

- Team input: this is paramount, and most people left it out. What does the team want to do? Is this something they'd like to learn? Who would want to go to training, if offered?
- Bus factor: How will the team avoid this?
- Business value: Is this something that may be valuable to future projects? If so, the team may want to become the company experts, and the company may be willing to invest in making that happen.
- Focused exploration: Let the project drive the use of the technology. Is there an early technology spike that would demonstrate viability for this project and its deployment context?
- Expert's role: Have the team consider alternative roles for the expert, e.g., bring the expert, if needed, after the team has done some AJAX coding, so that (1) time is not wasted on easy stuff (2) the team can ask better questions (3) the expert can critique what they've done.

A common mistake was being very prescriptive and simply telling the team what to do. This would be bad, even if you were there and knew more about what was going on. It certainly doesn't make sense with so little to go on.

Project start

Kudos to Katie Zhu for the "The Agile Samurai quote" you pick your architecture when you pick your team."

Grading: Did you focus on risks particularly likely for this project, and suggest specific steps to address them? Particularly big risks:

- No real users: almost everyone missed this. A lot of people talked about getting user stories but never said how. Almost no one told Smedar of the importance of identifying some departmental users who make the purchases. This is a big problem in many projects: the client is usually not the user, and is pretty useless for developing user stories and acceptance tests.
- Student developers: many risks here, e.g., limited time, conflicting schedules, variable skill levels, bus factor, and high turnover. Most people had good suggestions, like pulling tasks, very small tasks, cross training, etc.. The main issue was having enough ideas given the number of risk factors. Daily face-to-face standups however is impractical.
- No quality-control: Because of the developer issues, it's also important to recommend establishing some processes and technologies, like source control and CI servers, early on so that when new developers join, which is certain to happen, it's easy to get the current code and there's a well-established test-driven development practice.

I do not see deadlines as a particular issue for this project. Nothing indicated a date that had to be met. The lack of deadline can be a problem for waterfall projects, but not if there's an inherent push for as early a release as possible.

A problem with some responses was "a lot of do, not much why," i.e., listing a bunch of agile practices, with no adaptation to the situation and no explanation of why these practices are relevant.