a case study of post-deployment user feedback triage

Andrew J. Ko and Michael J. Lee, Valentina Ferrari, Steven Ip, and Charlie Tran

dub | Information School UNIVERSITY of WASHINGTON W

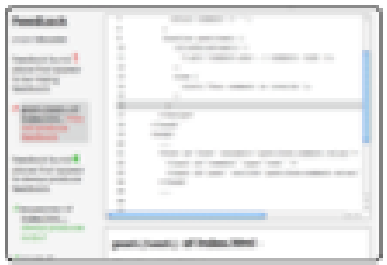
4th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), May 2011

Modified version
2011-11-10 Lutz Prechelt

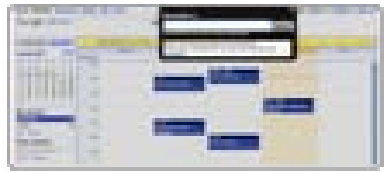
usegroup

BUGS

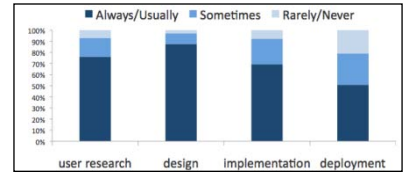
detecting

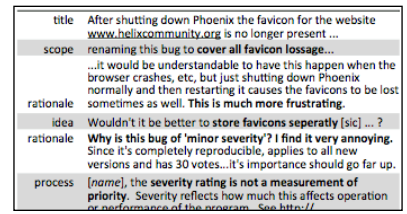


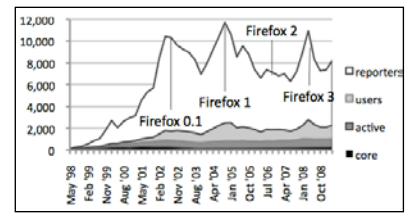
diagnosing

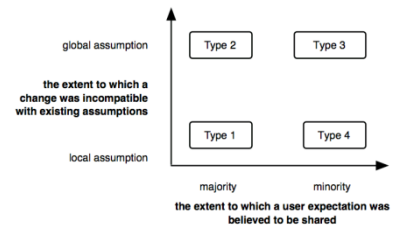


triaging

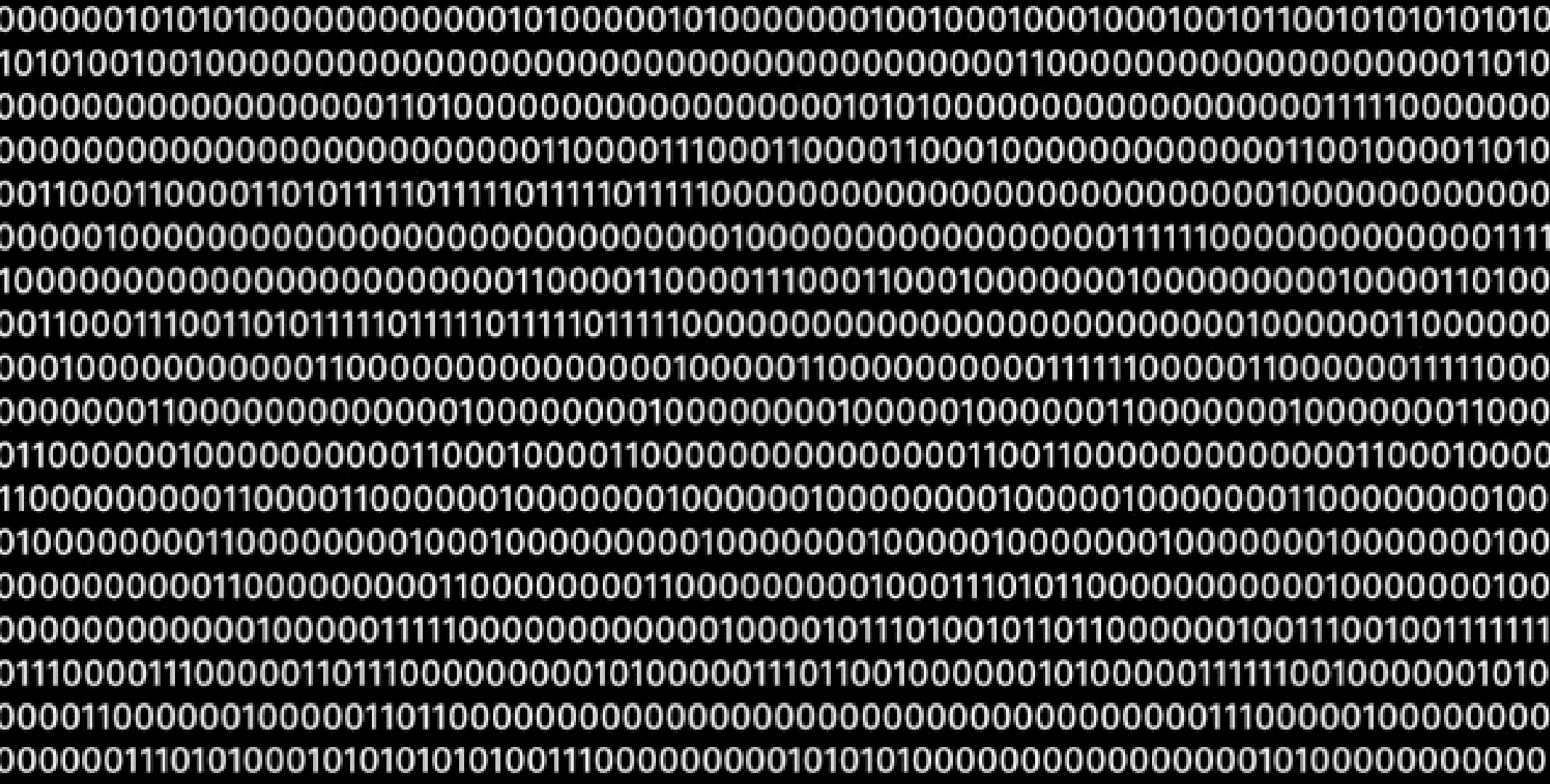








we use software to represent all kinds of information, people, processes



but it doesn't always represent things the way that people want...

cebook Search

he Facebook Blog

He/She/They: Grammar and Facebook.

by Naomi Gleit on Thursday, June 26, 2008 at 10:37pm

As Facebook grows in other languages, we are learning a lot about what the "Facebook Experience" is like for people around the world. One of the first challenges was getting words that are really long in other languages to fit on the screen properly. Recently, we've been figuring out how to deal with a new challenge—grammar.

Ever see a story about a friend who tagged "themself" in a photo? "Themself" isn't even a real word. We've used that in place of "himself or herself". We made that grammatical choice in order to respect people who haven't, until now, selected their sex on their profile.

However, we've gotten feedback from translators and users in other countries that translations wind up being too confusing when people have not specified a sex on their profiles. People who haven't selected what sex they are frequently get defaulted to the wrong sex entirely in

by representing human endeavors, software even *changes* what people want

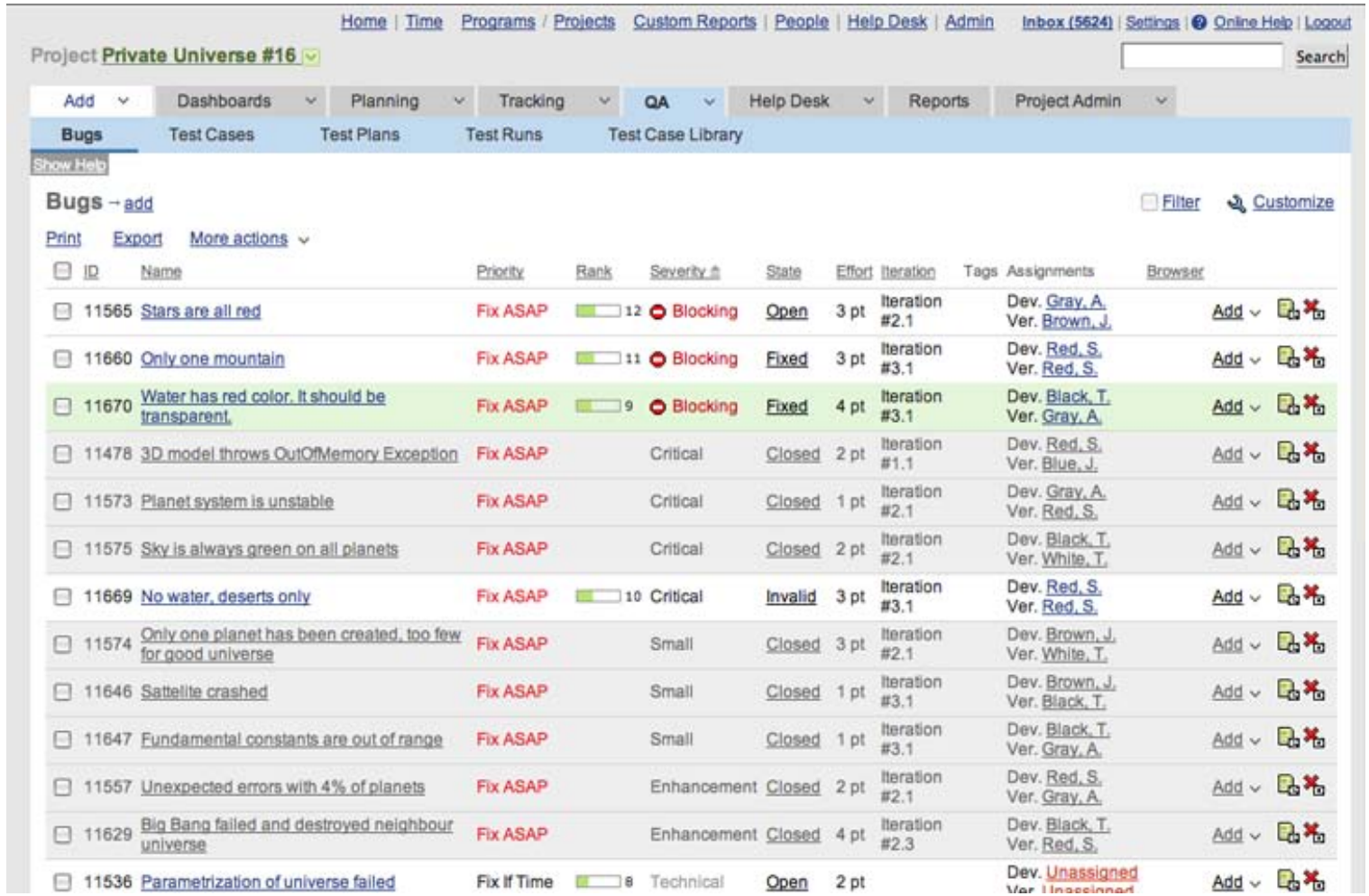
"People have really gotten comfortable not only sharing more information and different kinds, but more openly and with more people. That social norm is just something that has evolved over time.

We view it as our role in the system to constantly be innovating and be updating what our system is to reflect what the current social norms are."

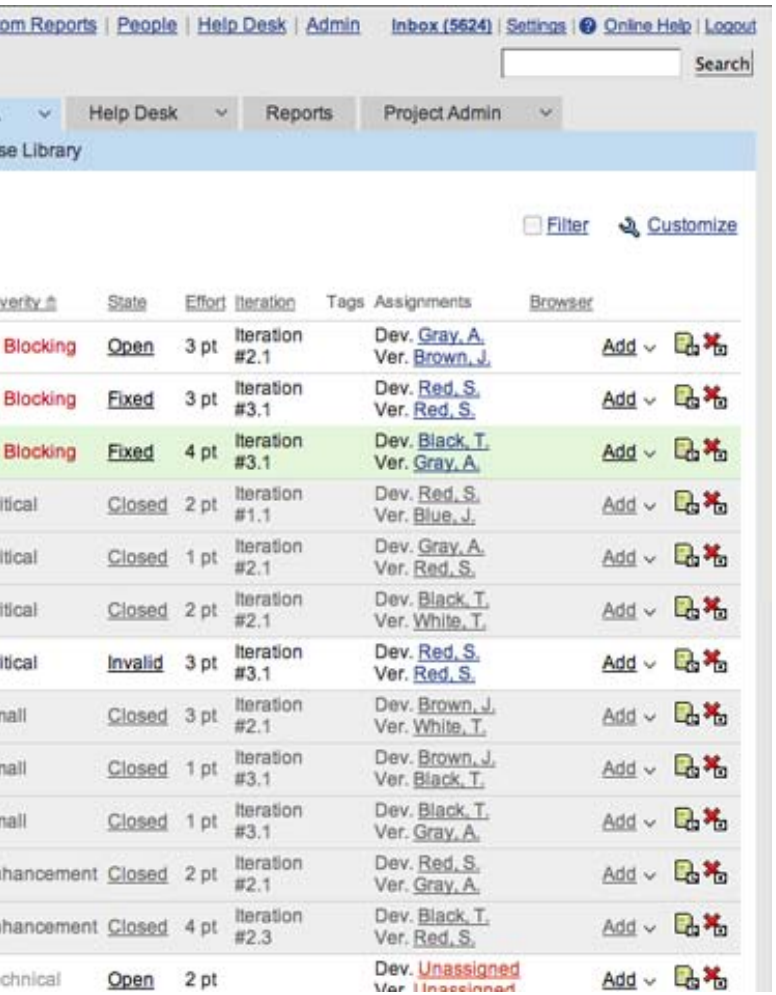
– Mark Zuckerburg

bug
triage
unintended
behaviors

feedback
triage
undesirable
behaviors



feedback
triage
undesirable
behaviors



how do teams triage user feedback about undesirable behaviors?

what constrains a team's response to feedback?

we performed a case study of **LST**

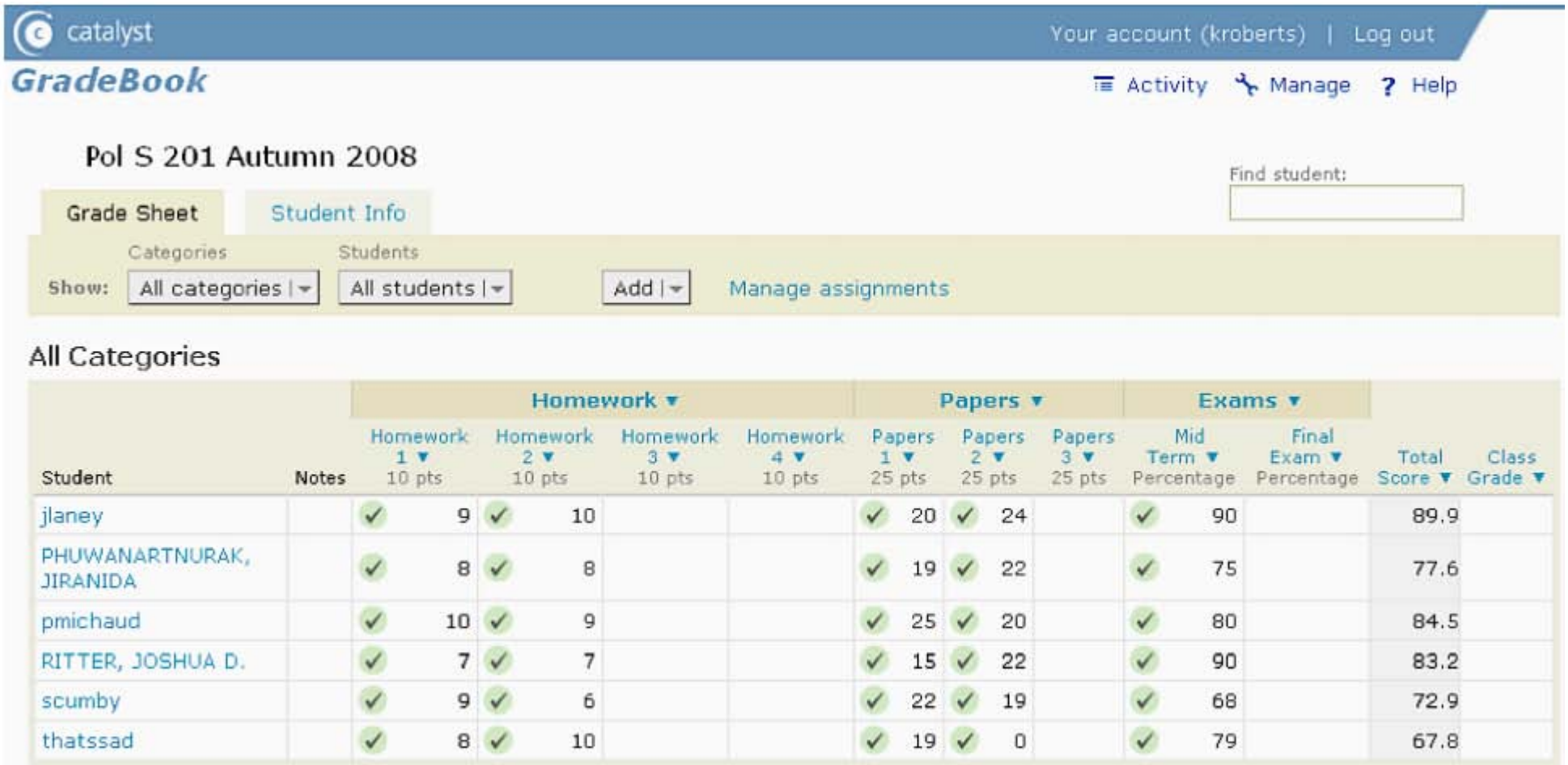
they develop a suite of 15 web applications to support teaching and research

their mission statement

“We follow an iterative, user-centered design and development process that focuses on understanding the needs and experiences of our users. Whether we are creating a new tool or updating an older one, our design decisions are based on direct feedback, user research, and findings from usability studies.”

The GradeBook project

- Used by university instructional staff to store, organize, and publish student grades.



project history

2 full time developers,
3 designers,
1 managers

project started because the community wanted to move beyond paper grade submission

user research began in 2007, including

interviews with dozens of faculty, TAs, students, staff

surveys of thousands of students about grading experience

content analysis of faculty grading spreadsheets and paper grade books

project history

2 full time developers,
3 designers,
1 managers

6 months of prototyping and participatory design with key informants

launched in September 2008

our interviews and observations began September 2009 and lasted 6 months

in the 1.5 years since launch, the team received tens of thousands of help requests and filed 1,490 bug reports

the team’s process

2 full time developers,
3 designers,
1 managers

daily morning scrum meetings

biweekly sprint planning meetings

developers, designers, and PM all attend, PM leads

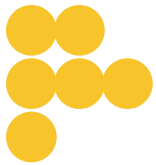
the methods we used

2 full time developers,
3 designers,
1 managers

- 6 months of field observations by the 2nd author
- at the help desk, amongst the support staff
- at the development office, observing sprint planning, triage, testing practices, and developer collaboration

the methods we used

2 full time developers,
3 designers,
1 managers



two 2-hour semi-structured interviews with 4 of the 6 GradeBook team members focused on rationale for project

details about user research and prototyping performed

probed into aspects of the software that the team wanted to evolve in response to user feedback, but could not

the methods we used

2 full time developers,
3 designers,
1 managers

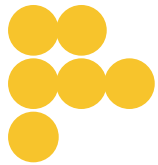
10 semi-structured interviews with 12 instructional staff teaching lower-division undergraduate courses

focused on course syllabus, grading practices, grade storage, feedback delivery, use of GradeBook

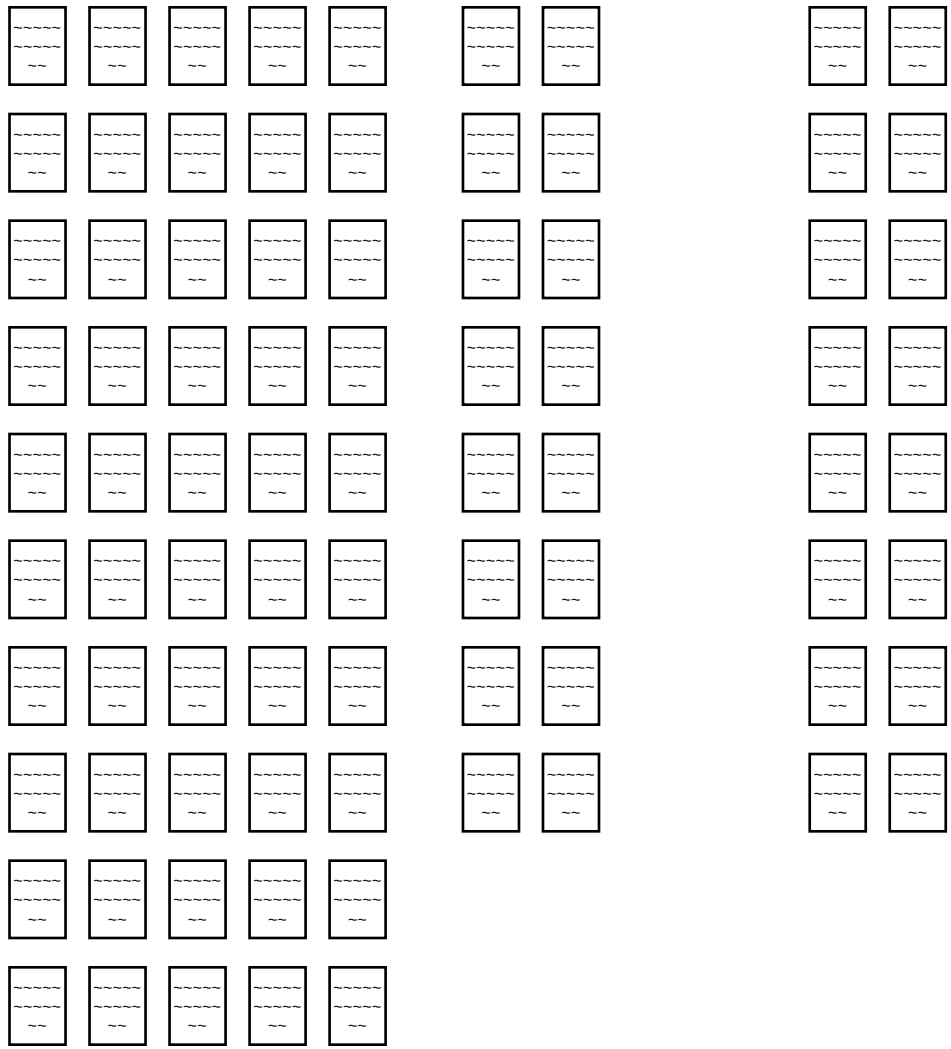
qualitative analysis of 1,490 FIXED and WONTFIX bug reports across the history of the application

the methods we used

2 full time developers,
3 designers,
1 managers



analyzed the decision rationale in the bug reports, interviews, observations

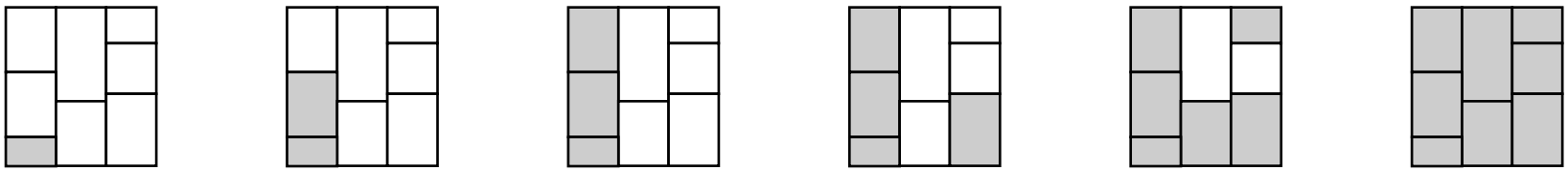


for each issue, the team focused on two major questions

for any given desired behavior...

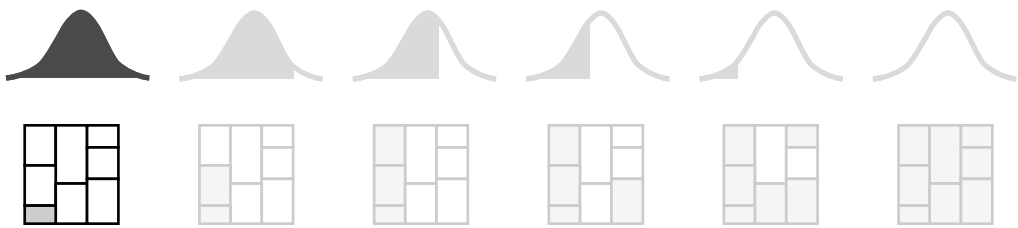
how many users desire it?

how much of the code must change?



let's discuss four examples...

majority expectations, **local** assumptions

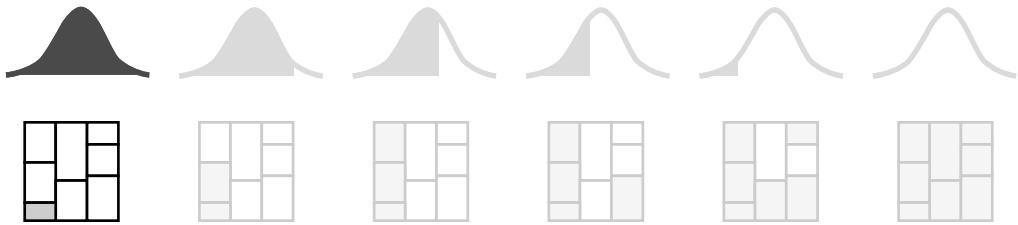


the easiest kind
of change to
implement

e.g., a modal dialog box that confused everyone, helped no one, and required the change of a single Boolean in the source code

e.g., an uncaught error condition that confused everyone, helped no one, and required the addition of a guarding conditional and an error message

majority expectations, **local** assumptions

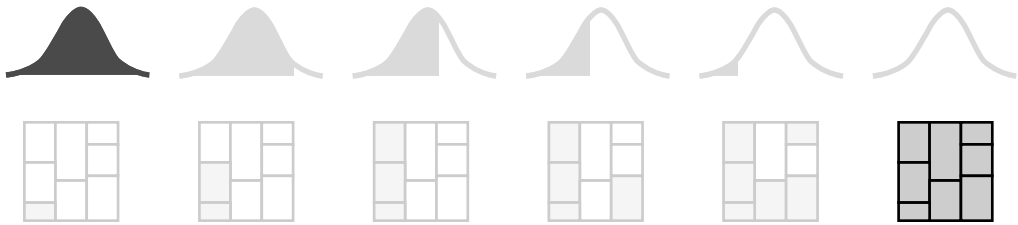


not always something that should be implemented

unambiguous computation of grades uncovered
unconventional conceptions of grades

“We didn't initially support 4.0 scale scores. And this has been, its really a pedagogical debate, in some ways... A lot of faculty want to use 4.0 scale grades for all assignments in their class and then do calculation on those. And the software says, "those aren't actually real numbers, those are more like a ranking," because its not a literal scale from 0 to 4. But trying to communicate to faculty who've been doing this for years in Excel and thinking there's absolutely nothing wrong with it is really difficult.

majority expectations, **global** assumptions

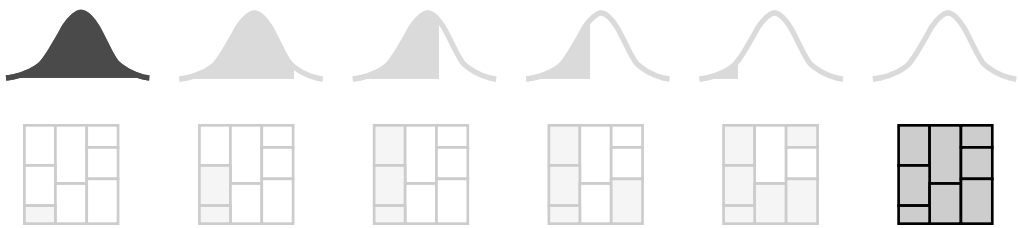


difficult to change but important to most users

e.g., the original implementation used the concept of a ”group”, borrowed from existing tools, and added the concept of a “class list”. Users had to create a group that contained the class list in order to get the privacy functionality of a “group.”

→

majority expectations, **global** assumptions

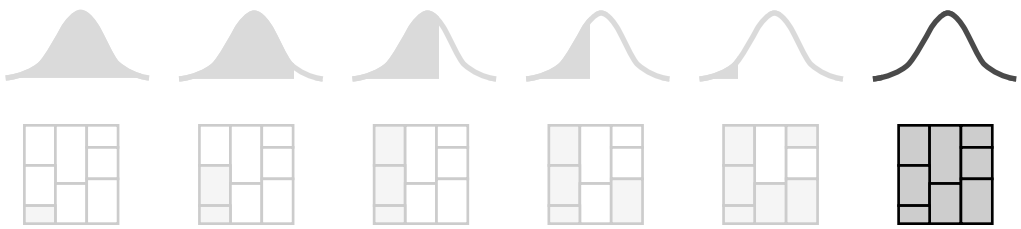


difficult to change but important to most users

merging these two concepts would have caused a major rewrite and data migration, so they initially hid the complexity behind the user interface, but this hack rippled through the system, surfacing in other user workflows.

“That was a huge data migration process and it caused a lot of pain, and that was probably like 3, 4 months of time... We ended up paying for it later on when we had to undo that work... we had to go in in GradeBook and change all the code that was making that assumption for us, and remove the ad hoc group from ever being created, because we didn't need it anymore...”

minority expectations, **global** assumptions



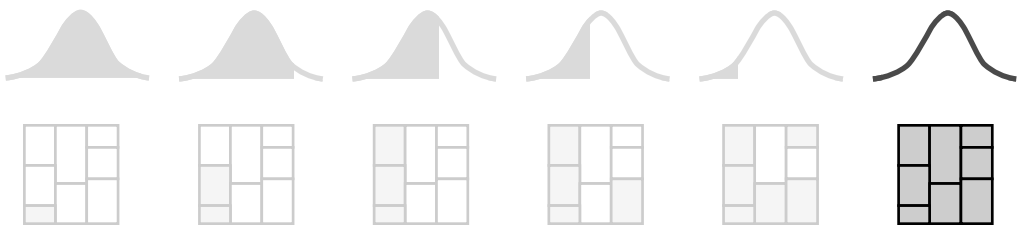
difficult to change and a minority concern

e.g., the team had focused on testing the performance of the application on class sizes up to 40 students with a few dozen assignments

suddenly, however, it was so easy to track assignments that some instructors began adding columns for every little bit of participation credit, which they used to track on paper

For them, performance for even small classes became a major issue →

minority expectations, **global** assumptions



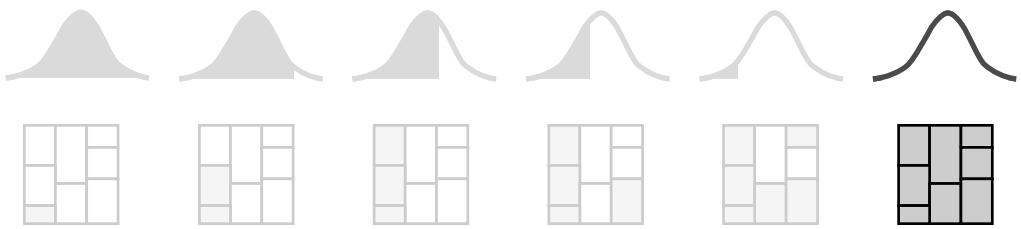
difficult to change and a minority concern

and yet still, this was a minority of instructors

improving performance, however, was limited by the choice of grid view, upon which was built many other user interface features

replacing it with something with greater performance would have required a new user interface implementation

minority expectations, **global** assumptions

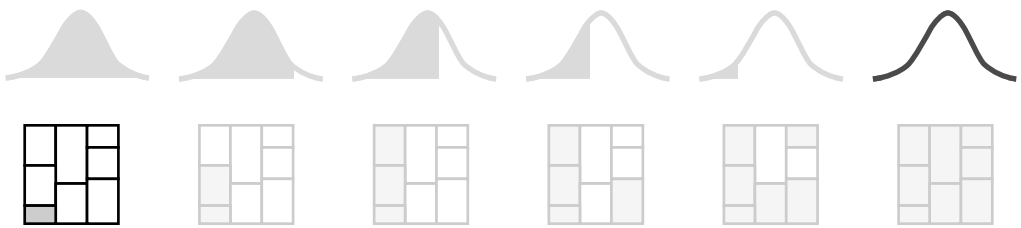


difficult to change and a minority concern

the team perceived this to have led to lower adoption

“Unfortunately, I think a tool gets released, they check it out, and then they go, oh, its too slow. Okay, well we hear that and we fix it, but if your first impression of the tool is that its too slow, its not a whole lot to bring you back the second and third time.”

minority expectations, **local** assumptions



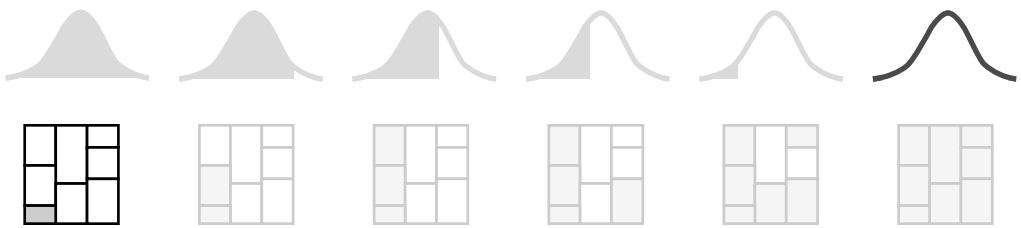
easy to change minority concern

some requested changes were very minor implementation changes

e.g., in one case a user pointed out that “X” was a valid grade, but when importing an Excel spreadsheet with an X grade, GradeBook marked it as invalid until the user explicitly selected “X - No grade now”

→

minority expectations, **local** assumptions

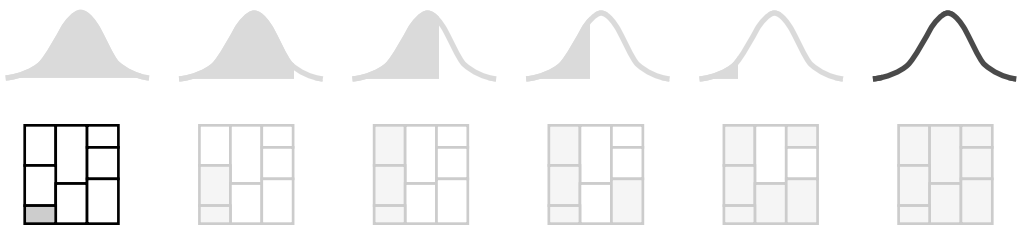


easy to change minority concern

the user asked for automatic conversion, but this would make unsafe assumptions:

“I think this concern is bogus (to be pedantic X - No grade now is not even a grade), and transforming a 'X' to 'X - No grade now' seems like a big leap to me... We want because we want to be sure he's gone through them and specifically assigned an "X" or an "I" and that it isn't some mistake. The other factor that is causing this is that he is not really a GradeBook user, but someone trying to import grades at the end of the quarter for the sole purpose of submitting...

minority expectations, **local** assumptions



easy to change minority concern

some concerns were high severity, but low frequency, and treated out of scope

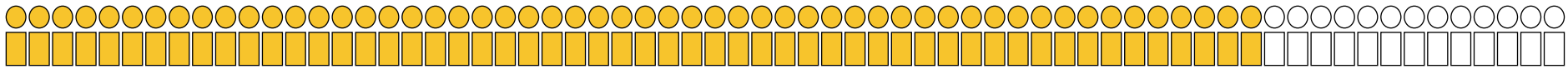
> She needed to submit a final grade for one student within 2 hours, because the student's financial aid was depending on it. However, she had 30 other students that she wasn't ready to submit... This puts her in a very sticky situation...
>
> [wontfix] The registrar does not let you do such a thing. That's why there's the X (No grade now). Unfortunately that is not much help to this instructor, but that's the way it is for now.

use restricts change

the primary constraint on evolving GradeBook was the **majority need**

addressing a minority need often meant breaking the majority's use

therefore, designers must continually decide who the software serves



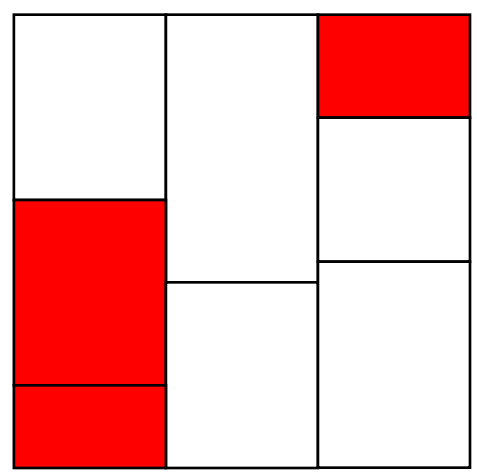
and who it does not

architecture and anticipated use are often misaligned

modularity (and thus flexibility) was aligned with other goals, such as performance and coordination requirements

which changes were easy to implement seemed **accidental** rather than **intentional**

this was easy to change, but didn't need to be



this needed to change but couldn't

the implementation

implications

are there ways of architecting software to **defer** design decisions about unknown global expectations?

for example, perhaps the GradeBook team could have implemented a schema that would account for many possible future changes to representation of grades

could the level of abstraction and modularity in data schemas be **proportional** to a team's lack of confidence in its user research?

align flexibility with anticipated use

implications

how well do software teams actually **know** the majority use?

many minority reports could have appeared to be a minority view, but maybe in reality represent a widespread but underreported problem?

there was a general belief that the only problems that mattered were the ones users reported

implications

"software evolution" is an apt description of GradeBook's change over time

change was **slow**

users were slow to adapt

change was **punctuated**

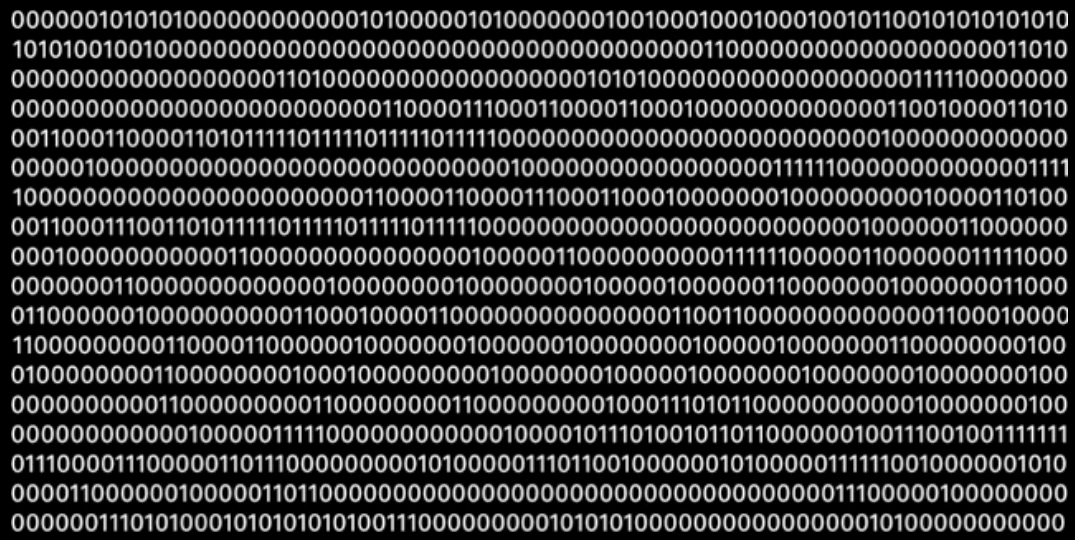
e.g., data migrations

major changes came only when the software was introduced to a new **environment**

e.g., registrar mandated electronic grade submission

takeaway...

software change is constrained by misalignments between **anticipated** and **actual** use



questions?