Post-mortem

CIS 322
March 14, 2017

Today we're going to talk about reviewing and learning from past projects.

Everyone should be able to write strong course evaluations after this class. ;)

From the Dailies

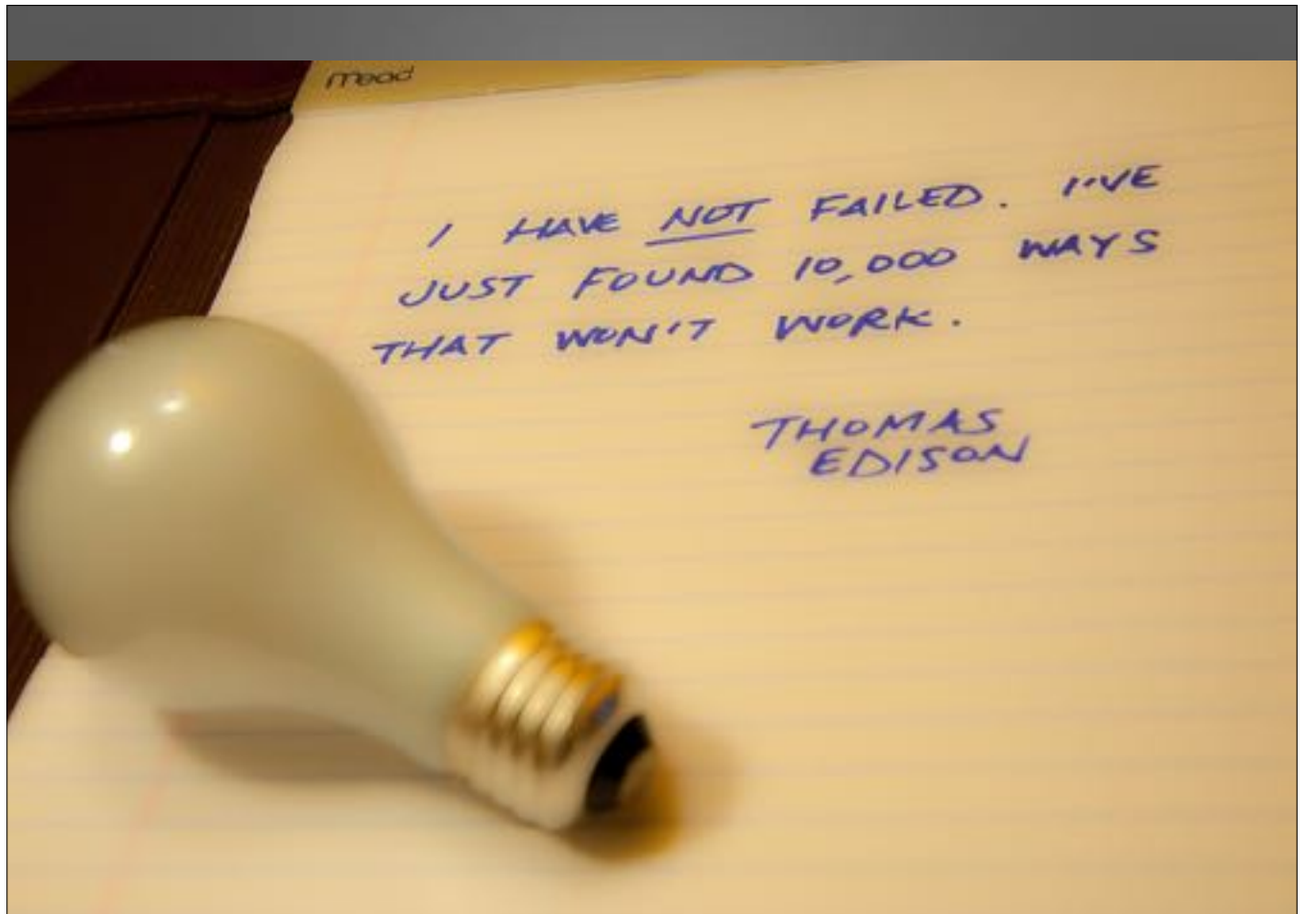
- Data export was not difficult overall, since I know the structure of my own database. Import was a bit more involved, but still rather simple. Especially so compared to the data migration assignment early in the term. Strict formatting procedures are next to Godliness.
- I figure I can make tests even though parts are not going to work.
- Finished my test plan document and uploaded it to my assignment 10 branch. Also have notified my partner so that they can start testing as soon as they'd like. Will start the web clients tomorrow or Wednesday.
- I just wanted to clarify on the final project... Looking at the Rubric it appears that it will be just all of our completed homework assignments. Is that correct? Are there things we would need to add? For example, is Revoking a user new? (I don't remember it being in the previous homework, but that doesn't mean it wasn't there.)

I think the general sense is that last week's data migration assignment was better than the one earlier this term.

Please start on your test document early. Test documentation can and should be written before system delivery. The tests don't need to pass to get credit for generating/executing the test plan. To a certain extent it is more valuable to know that tests are failing.

Yes, the intent has always been that the final project would be the system you would have if all of the weekly assignments were done correctly and integrated correctly. The coding portion this week and next are intended to be lighter so that you have time to fix defects and do more testing of your code before the final project is graded. Revoke user is part of the week 10 assignment, where the web services are added.

Andy has the lecture on Thursday and will be covering web services versus screen scraping.



https://www.flickr.com/photos/8047705@N02/5366637592/in/album-72157625724176990/

School teaches us real unhealthy ways of relating to not being perfect on the first try. Which is doubly terrible since we learn very little when things work right. Failure is really the only thing we can learn from, our scientific methods don't prove things right. To be testable there needs to be a way to falsify it.

Every project you are ever on will be messed up. It might not be a catastrophic failure but many many things will go wrong during the lifetime of the project.

The most important thing to do with experience is to learn from it.

Some recent incidents…

- Feb 28, 2017 - AWS
  https://aws.amazon.com/message/41926/
- Jan 31, 2017 - GitLab
  https://about.gitlab.com/2017/02/10/postmortem-of-database-outage-of-january-31/
- Collection of postmortem links:
  https://github.com/danluu/post-mortems

There are lots of postmortems available for operational incidents. Why do you think this is? I would guess this is because operational failures are a lot more visible than failed development activities. After dependency has been built on a system failure costs big money, how can you guarantee that it won't fail again if you can't identify the root cause of the failure.

A project, program, or account manager should probably run this process…

To understand what really happened on a project we need input from all of the people involved in the project. We assume everyone wanted the project to be successful.

Succeed or fail, the end of a project tends to be an emotional time and discussion is colored by what the final result was. To avoid repeating failures and improve success we need to understand what the situation was at the time the decisions were made. Emails, status messages, meeting minutes, commit logs can all help with this.

These artifacts should help to establish a timeline for what happened during the project. The timeline helps to track cause and effect relationships and can highlight where miscommunication occurred… Having the whole picture it can be clearer to see everyone making their best local choice led to the end experienced.

At the end of the day, our projects are about people working together. Everyone needs a chance to share their perspective on what happened. The language should focus on the technical facts of the project, when language slips in for finger pointing and blame the value of the exercise is lost.

Just by completing the exercise the participants will be better at future projects. Generating some documentation to improve organization processes is even better. If something worked really well, let's understand that more and try that more often. If something failed miserably, let's understand that and protect ourselves from making that mistake again.

SMART Goals

* Communicate project status and risks
* Effectively use Git, python, bash, etc
* Write a full stack web application
* Reason about solution designs

Let’s look back at what we set out to do this term… This sequence of slides was from the first lecture.

What are we going to be able to do by the end of this course?

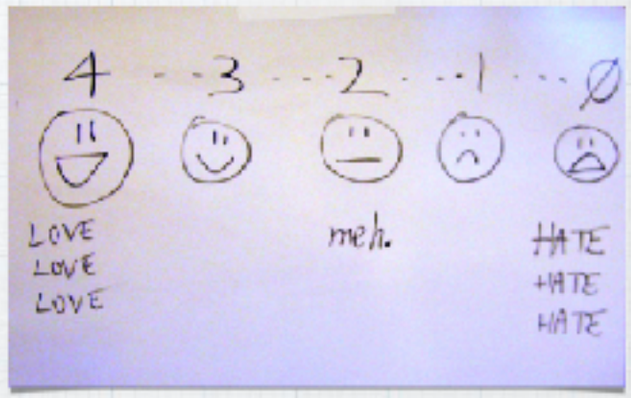
Approach

* Regular Lecture + office hours
* Implementation of a full stack web application
* Simulated management

How are we going to do that?

Evaluation Plan

* Term Project
* Incrementals
* Code Reviews
* Weekly Status
* Daily Checkpoint
* Tests



How are we going to evaluate that we are being successful at that?

No Plan Survives Contact with Reality

* Be self aware
* Start early
* Get help early
* The sooner we know, the more we can do

Right out of the gate there was a huge hedge in the project… I've done enough projects to know that sometimes things go wrong and that major changes are needed to correct. I'm glad we had the information and flexibility to make the needed change mid-way through the course.

These are the goals Andy and I captured from y'all during the first lecture. Based on your experience in this course, have we succeeded or failed in achieving the course goals?

Stop Light Chart

	Daily status messages. This significantly increased visibility into project progress and facilitated course realignment before crashing to badly.
	Virtual machines solved some problems but introduced others. Hosted VMs might have solved many of these problems.
	Monday morning due dates. Induces a lot of weekend hours and supports late starts. OTOH, the weekend makes balancing the work easier for some students.
	Assignment specification at the level of business requirements is too hard (for both implementation and grading). Intermediate representations are needed for assignments.
	Organizing lecture like a dev meeting. A major failure due to sidedness of lecture format and contribution structure.

A simple document we can produce out of a project postmortem is a stop light chart. Execs like them because they look simple and have colors… letting the exec feel like they understand what is going on when even when clueless.

Green - For the things that were good
Yellow - For the things that were ok but could’ve been better
Red - For things that should be avoided in the future

Lessons Learned

- Lack of a text book is problematic.
  Not having a structured source puts significant pressure on lecture and requires unsustainable energy in course material development.

- Branches and tagging need to be week 1 topics.
  Using version control for assignments can be very good but which revision to grade quickly becomes a challenge. Tags or branches could solve this issue

- Automated testing harness would've made life better.
  A way for students to test submissions would build/install would have significantly reduced grading time costs

Lessons learned lists are like stoplight charts but less colorful.

Actions

- Do not accept teaching responsibility with out a supporting textbook in mind or provided
- Be more aggressive about completing the complete assignment suite before class starts and keep records on implementation time for assignment tuning
- Identify VM hosting solution
- Add automated testing hooks to DERP to at least verify solutions install

Ultimately, what are the things that we change in our behavior and future actions to have a better future.

LOST Postmortem Activity

What worked? What didn't? How would we do this differently?

For the rest of lecture, I'd like us to talk through what worked and what didn't on the LOST project and this class.

This course has been challenging with a significant workload. You've all done well keeping up with the delivery pace and have built some really excellent software given the tight timeline.