Lecture 28: Finale

CS 5150, Spring 2022

Course reminders

- Final report due tonight (Gradescope)
  - Handoff package committed to repo/wiki (will take snapshot tonight)
    - Please include final presentation slides
  - Working code merged to master/main branch (resolve conflicts)
  - Work with client on testing deployment
- Peer evaluations for session 5 (CMS)
- Course evaluation due Fri
  - Homework point
- Instructor office hours continue this week
  - Happy to discuss software engineering more generally

Lecture goals

- Recognize ethical dilemmas in software projects
- Know when to cancel a project
- Celebrate project achievements

Professionalism & Ethics

Poll: PollEv.com/cs5150

What should you do if you discover a major security vulnerability in a piece of widely-used software?

Responsible disclosure

- AKA "coordinated vulnerability disclosure"
- Coordinate timing of announcement with vendor
  - Give them time to patch products, prepare press response
  - Upper bound on timing to hasten vendor action (typ. 90 days)
- For open-source projects, look for security policy (SECURITY.md)
  - Contact Vulnerability Management Team or owner
  - Do not post details to public mailing lists, chat rooms
- May be assigned placeholder CVE to coordinate efforts without disclosing details

Poll: PollEv.com/cs5150

Which of these development efforts would you be comfortable contributing to?

- Drug marketing campaign
- Click fraud
- Selling 0-days
- Reverse engineering
- Weaponized AI
- Selling personal data
- Bitcoin mining

Ethics

- Software can harm society beyond physical injury
- Personal fulfilment is important too
  - Take responsibility for your work
  - Avoid future regrets

- Compared to traditional engineering, software has *less oversight* and *wider impact*
  - Amplification: One day's work can affect millions of people, consume millions of hours

2009 HP webcam

https://youtu.be/_YOoukA_Kp8

How bad is this?

- Just a buggy, experimental gimmick for consumers?
  - Tracking faces with dark complexion *is* more technically difficult
- Demeaning to users, especially if systematic
- When widespread, denies some users full participation in modern society
- What about biometric authentication?
- What about criminal suspect databases?

Diversity

- Wider impact => more diverse user base
  - => More potential to reinforce stereotypes, inequity
- Failure to anticipate/respond to biased systems can lead to major societal (not to mention reputational) harm
- Need to expand diversity during development (shift left)
  - More diverse developer teams
  - More diverse user testing
- "Single source of truth" does not apply to human society
  - Disputed borders
  - Different interpretations of words/phrases/symbols
  - Different value systems

Ethics extends beyond code

- Hiring practices
  - Beware affinity bias, groupthink
- Promotions/opportunities
  - Beyond mentoring - advocate for coworkers who do good work but seem to go unnoticed
- Decision-making
  - Don’t defend decisions solely on precedent
  - Look beyond direct “bottom line” impact

ACM Code of ethics and professional practice

1. PUBLIC – Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER – Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT – Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT – Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT – Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION – Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES – Software engineers shall be fair to and supportive of their colleagues.
8. SELF – Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Examples

Causes of poor outcomes

- Normalization of risk
  - Space Shuttle Columbia
- Over-constrained compute resources
  - Numerous space systems
- Over-trust in inherited components
  - Ariane 5
- Discounting cost of “inert” or “extra” components
- Changing circumstances
- Poor client-dev understanding

Ask for help

- University team given government funds to build high-performance gateway
- Promising young developer hired, assigned task
- Task too difficult, but he hid his problems for months
- Project cancelled, nothing delivered

- Don’t try to maintain a reputation at expense of project
- Asking for help is expected, helps team grow
- Leaders must monitor new employees more closely

Know when to cancel

- Senior management (without consulting technical staff) decides to replace administrative software with COTS solution
  - Adopted schedule and budget from vendor's marketing (hopelessly optimistic)
- Staff became dispirited; many left, including CIO
- What should new CIO do?

- Analyze situation, provide visibility to leadership
- Identify work worth continuing
- Cancel remainder of project

Know when to start over

- University working on a joint project with a company to develop new system software
- After two years, junior developer convinced university leader that technical approach was wrong
- University decided to start over, company decided to keep going
- Both finished around same time, university version was superior

- The best time to refactor is before the system is first deployed

Project summaries

Takeaways

- Learning to navigate a large codebase is hard
  - Look for similar changes in its commit history
  - Trace execution of familiar functionality
- Avoid silos
  - Shared sense of responsibility
- Accountability
  - Explicit team expectations (including schedule)
  - Concrete deliverables
- Schedule estimation is hard
  - Use modeling, mock-ups to elicit detailed requirements early
  - Leave plenty of buffer for changes after testing (including deployment)

Successes

- External clients happy
- 11 internal projects achieved MVP
- Shoutouts
  - Team 10 (Gerrit)
  - Team 17 (Review Board)

Conclusions

- Software engineering is bigger than programming
  - More stakeholders
  - Collaborative development
  - Quality has a cost
- Successful projects involve tradeoffs, communication
  - Different projects warrant different approaches
- Big projects *are* possible
  - With planning & teamwork, can accomplish far more than solo

Good luck with all your future endeavors!