Lecture 2: Projects and process

Variety

Software must serve many different purposes, in many different settings, for many different people.

Project variety demands developer versatility.

There is no “best” way to create software in all cases.

A software engineer must know a wide variety of methods & tools and select appropriate ones for the project at hand

Stakeholders

First step in any project: identify the stakeholders.

Stakeholder interests are not always aligned.

Developers

- Should be considered a stakeholder (but not the most important one)
- Are also an expensive resource
- Have responsibilities of competence, confidentiality, compliance

Client

- Provides resources in exchange for software
- Bears risk in event of project failure
- Sets requirements, priorities
- Client satisfaction is primary measure of project success

PollEv.com/cs5150 Who is the client for general-purpose software products?

Customer

- Buys the software or selects it for use by an organization

User

- Actually uses (interfaces with) the software

Society

- May be affected by the software
- Often not represented when stakeholders are consulted

- Advisable to appoint an *advocate* for their interests
    - Automated processes tend to become invisible
- Risks to society should be identified and acknowledged

Breakout

3 minutes

1. Note your breakout room #
2. Select a reporter
3. Identify the stakeholders (developer, client, customer, user) for:
    a. canvas.cornell.edu
    b. The FAA's Advanced Automation System (based on assigned reading)

Risk

- All projects require tradeoffs between **function**, **cost**, and **time**
- Who should set priorities when deciding tradeoffs?
    - The client should be given the information necessary to make an informed decision based on their priorities

Consequences

- Failed projects can bankrupt companies.
- Managers can lose their jobs
- Users and society may be harmed

Minimizing risk

Communication

- As much as half of delivered software is never used
    - Developers build the "wrong software" – doesn't meet client's needs
- Developer must work to understand client, customer, and user expectations
- Developer may add technical insights, but client satisfaction is the primary measure of success

Communicating via deliverables:

- Feasibility study
- Requirements and design (separated)

- Milestones & releases
- User & acceptance testing
- Handover

Visibility

- Those responsible for the project (client, managers) must know what is happening
- But most developers are bad at providing good visibility
- Large projects dilute visibility at each layer of hierarchy
- Working software provides good visibility
    - But be upfront about limitations

Short dev cycles improve visibility

- Risk accumulates with time since last check-in
- Deliver working software frequently (weeks rather than months, or even continuously)
    - Clients, customers, & users can evaluate work
    - Opportunity to adapt to new circumstances

Management

- Project management
    - Track progress against schedule
    - Prioritize tasks
- Personnel management
    - Allocate the right number of developers with the right skills at the right time
    - Ensure that developers have a productive work environment
- Compliance advising
    - Understand legal, regulatory, economic environment
- Development processes
    - Enforce best practices to minimize risk without excessive overhead

Development processes

- Composed of process decisions
- Tradeoff between risk reduction and overhead
    - Effectiveness, cost depend on tool support, developer skill, culture

    - Initial risk depends on project size
    - Risk tolerance depends on application
- Must adapt process to each project
- Aim to improve processes throughout project

Process steps

- Project specifics are different, but they need to address similar issues
- Process decisions should be adopted to address common process **steps**
- Testing & documentation occur in many steps

- Feasibility & planning
- Requirements
- System & interface design
- Program development (includes program design)
- Acceptance and release
- Operations and maintenance

Breakout

3 minutes

1. Note your breakout room #
2. Select a reporter
3. Discuss: Which process steps were handled poorly for the FAA's Advanced Automation System? (based on assigned reading)

Software methodologies

- Formal vs. informal (output of steps)
- Duration and ordering of steps

Heavyweight:

Lightweight:

Waterfall model

- Heavyweight
- Based on traditional engineering project management

- Was a reasonable choice for early software
  - Well-understood requirements
  - Standardized system design
  - Coding was tedious
- Advantages
  - Separation of tasks
  - Process visibility
  - Quality control, cost monitoring at each step
- Disadvantages
  - Does not account for revising earlier stages
  - Not flexible enough to react to changing conditions

Iteration is required

- Work in later stages improves understanding of considerations in earlier stages
- Context, requirements do change over course of development

Cost of defects

- Cost of defects can be exponential in time between introduction and discovery
- "Shift left": catch defects as early as possible
- Traditional waterfall deliverables are not effective at catching many defects

Modified waterfall model

- Appropriate when requirements are well understood and system design is fixed
- Recommended for safety-critical and highly-regulated systems
  - Requirements must be thoroughly analyzed and documented

Iterative refinement

- Medium-weight process
- Create early prototype
- Review prototype with stakeholders; clarify requirements
- Refine prototype
- "Plan to throw one away" (prototypes are not release-worthy)

    - Prototype can't be used to evaluate non-functional requirements
    - Revising working but low-quality code is difficult; better to start over after requirements are clarified

Incremental delivery

- Deliver fully-tested increments with subset of functionality

Agile methods and eXtreme Programming (XP)

- User stories
- Incremental planning
- Small releases
- Simple design
- Test-first development
- Periodic refactoring
- Pair programming
- Collective ownership
- Continuous integration
- On-site customer

Often not realistic as originally proposed. Relies on developer autonomy.

Scrum

- Provides management structure for XP
- Work scheduled in time-boxed "sprints" (2-4 weeks)
- Tasks selected from backlog
- Sprint product is production-ready code + docs
- Daily team meetings
- Benefits
    - Provides good visibility and communication
    - Accommodates vague, changing requirements
    - Popular for small, dynamic projects
- Challenges
    - Tricky to scale to large projects
    - Poor fit for bureaucratic organizations
    - Works best with highly-skilled developers
    - Hard to validate requirements for completeness

    - Lack of formal “internal” documentation impedes maintenance, handoff

Integration and configuration

- When system design is standardized, can better take advantage of code reuse
- Leverage commercial-off-the-shelf (COTS) platforms
- Integrate and configure existing components based on client requirements
- Reduced cost and time, at expense of functionality

PollEv.com/cs5150 What methodology was used for the FAA AAS? Was this an appropriate choice?

Mixed processes

Many projects mix elements of multiple methodologies

Phased development

- Early benefit from initial investment
- Clarifies requirements for later phases
- Facilitates cost control

Summary

- Different development processes are appropriate for different projects
    - Allow for process evolution and improvement
    - Include all process steps (and keep them straight)
    - Accommodate revision of prior steps
    - Beware buzzwords
- Risk-reduction practices
    - Prototyping
    - Frequent releases, or decomposition into phases
    - Early & iterative testing with stakeholders
    - Promoting visibility

Next up

- Teamwork and collaboration tools

CS 5150 SP22

- Assignment: Read *Software Engineering at Google*, Chapter 2: How to Work Well on Teams