CSE 481 – Software System Design

Class goals	5 minutes
Reflections from past courses	10 minutes
Class organization	5 minutes
Project organization	10 minutes
Viable proposals	40 minutes
Schedule slip	10 minutes

"Real" Software Development

A "real" software development effort involves at least:

- a client(s)
- architecture, tool, platform, component base, project management, and other decisions
- a significant team consisting of developers and other contributors
- a schedule/budget
- programming
- a test/accept methodology
- documentation
- a deliverable

Our Curriculum (Projects)

- No client
- *Decisions* largely made by the instructor
- Small, developer-only teams
- A schedule consisting of the due date; no budget
- *Programming*
- Haphazard testing
- No documentation
- No (client) deliverable

The Games Version of 481

- No client
- *Decisions made by the dev team* (but restricted by the game engine)
- *Somewhat larger, more diverse teams*
- *An extremely informal schedule; excessive budgets*
- *Programming*
- Haphazard testing
- No documentation
- No deliverable

This Version of 481

In a way, all that is irrelevant.

Our goal: produce a useful system within budget.
(A given: The system is too large to be produced on time by an individual.)

Our other goals: gain first-hand experience with the problem, and processes that help address it; do the best job we can this time around, and learn how to do it better next time.

(Another given: We will make mistakes.*)*

Why Should You Like This?

- Why do you like programming? Many of the same characteristics are shared by this effort.
  - You're building an artifact.
  - You're creating an abstract structure.
  - You're solving problems.
  - It has both quantitative and qualitative aspects to appreciate.
- "Coding is $1/6^{th}$ of the project."

CSE 481 – Software System Design

Class goals	5 minutes
Reflections from past courses	10 minutes
Class organization	5 minutes
Project organization	15 minutes
Viable proposals	40 minutes
Schedule slip	10 minutes

Experience Says…

You're likely to relate to the projects exclusively in terms of programming.

- The "size" of the project will be estimated by the programming effort. Every other activity will seem like overhead.
- These projects are not so big that we need, or can afford, methodologies appropriate to very large teams.
- They're also not so small that they'll be forgiving of focusing exclusively on programming.

You're likely to want to build starting from a language, rather than from a set of existing components.

You're likely to want to directly code function, rather than to enable it (e.g., by building general interpreters, and putting the specifics into data).

Clients

"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is so difficult as establishing the detailed technical requirements, including all the interfaces to people, to machines, and to other software systems. No other part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later.

Therefore the most important function that software builders do for their clients is the iterative extraction and refinement of the product requirements. For the truth is, the clients do not know what they want. They usually do not know what questions must be answered, and they almost never have thought of the problem in the detail that must be specified. …"

Fred Brooks, The Mythical Man-Month

Depending on the project, the critical path may well be the repeated interactions with the client.

Communication

- Communication problems can break a team.
- Can be N-to-N or 1-to-N or anything in between.
- There's no substitute for face-to-face interaction.
- Coherent, written documentation affords resilience.

Schedules

- They slip.
- You won't know they've slipped unless you have one.
- A fuzzily defined schedule is next to useless. You need:
  - A complete schedule that takes you to the final product.
  - Estimates of the amount of work required for each component in the schedule.
  - Target completion dates.
  - A way to know whether or not you've met a target.
- Schedules (help) define priorities.
- Budgets, priorities, and schedules together help delimit the product goals.

Tools

- Brooks: *"It is worthwhile to build lots of debugging scaffolding and test code, perhaps even 50 percent as much as the product being debugged."*
- There are other tools, e.g., build environment, source control, programming language, component toolkits, format converters, install automation, documentation generators, open source projects, …
- Ask others for leads.
- There is (almost) no such thing as cheating in this course.

Get To "Integration" Early

"Integration" is a reasonably complete, skeletal version of the software.

- Helps decompose the task into relatively independent units ⇒ no one is held up by another.
- Provides a cushion for the deliverable – there's always a working version that is becoming incrementally more fully functioned.
- Provides a base for "system testing." (Integration problems can be among the costliest to correct.)
- Can serve as a "rapid prototype" for iterating with the client.

Reflections from past courses

Milestones

Have targets. Make progress in steps. Schedule the steps.

- Reach integration.
- Work to next milestone.
- Ideally, have some unambiguous way to assess when that milestone is met – e.g., "zero bug bounce" for code.
- Repeat

Reflections from past courses

Morale

- Morale has a huge effect on outcome.
- Outcome has a huge effect on morale.
  - *Get to integration as early as possible*
- A "sense of ownership" is critical.
  - *Everyone must have creative control over some portion of the project.*

Reflections from past courses

Day-to-Day

- Work steadily. 10 weeks is long enough to burn out if you try to sprint the entire distance. (Knowing where you're at helps – you need a schedule.)
- Make sure that what you're working on is a top priority item.

Reflections from past courses

Things You're Likely to "Forget"

Reliable >> On Time >> Function >> Pretty >> Performance

(Performance that limits function is a function issue.)

Convenient, low functionality >> Complicated, high functionality

Reflections from past courses

Other Things Worth Mentioning

- Know your audience. The more they have to learn to use your software the less likely it is that they will.
- Know your audience. The "right" solution may not work at all.
- Unexpected (runtime) events are expected. Plan for them.
- The software must be deliverable.
  - No two machines are the same.
- Licensing issues.

Reflections from past courses

CSE 481 – Software System Design

Class goals	5 minutes
Reflections from past courses	10 minutes
Class organization	5 minutes
Project organization	15 minutes
Viable proposals	40 minutes
Schedule slip	10 minutes

Class Organization

1. We won't do anything simply because "we should." Everything we do must justify its expense (effort) with its benefits.
2. At least initially, all decisions are up to you, except final project ship go/no-go (and possibly team membership).

Problem resolution:
- Try to empower each individual.
- Success and failure are both team efforts.
- The course staff is part of the team.

A Starting Point

Project spec:
- Tells team what they're building
- Tells client what they're getting

Architecture guide:
- Overview of conceptual underpinnings
- Defines interfaces
- May suggest implementations, tools, etc.

Schedule:
- Reflects our current best estimates
- Prioritizes day-to-day activities
- Raises warning flags
- Helps navigate when we go under/over budget

Assignments and Timeline

A first cut:
- This Thursday: architecture/requirements discussions; "final" project selections; roles assignments.
- Status meetings, attended by everyone, as needed.
- Wednesday, November 27: Delivery of beta systems.
- Wednesday, December 11: Delivery of final systems.
- Exam week: Retrospective critique during.

- Project status and schedule should always be visible to everyone.
- Individual status should be visible to everyone.

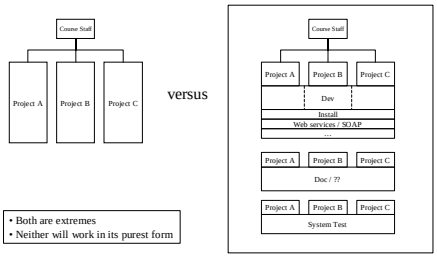
Grading and Workload

- Grade based on staff's impression of individual contribution.
  - Contribution to what?
- You don't have to work on this course 25 hours/week, but…
  - You're part of a team
  - It's important that others know what to expect from you
  - It's important to work to the success of the team
- Peer-evaluations
  - For feedback, not grading
  - Anonymous
  - During weeks 4, 7, and 10
  - Strengths and weaknesses

CSE 481 – Software System Design

Class goals	5 minutes
Reflections from past courses	10 minutes
Class organization	5 minutes
Project organization	15 minutes
Viable proposals	40 minutes
Schedule slip	10 minutes

Project Organization



- Both are extremes
- Neither will work in its purest form

Key Roles

There must be some per-project single points of contact:

- Architecture lead/interface
- Program lead/interface
- Documentation lead/interface

Each role is assumed by a single individual.
That individual will also function in other roles.

CSE 481 – Software System Design

Class goals	5 minutes
Reflections from past courses	10 minutes
Class organization	5 minutes
Project organization	15 minutes
Viable proposals	40 minutes
Schedule slip	10 minutes

Viable Proposals

Our goal for today:

- Choose about six candidate projects to look into further
- Choose 2-3 people/candidate to make a rough cut prediction of what it would involve
- Present/discuss those ideas on Thursday, and pick final set