PROCESSES
Unit Objective

• Understand the influences on a project
• Understand what a software process is
• Understand two common models
Every software project has three client controls:

- Cost
- Time
- Functionality

The tech team has three controls:

- Process
- People
- Technology

Software Engineering is about managing the client side and defining the tech side while managing risk.
Project Influences

- Most everything involves teams

- The effectiveness of the team relates directly to success

- Working with and within teams requires extra effort for
  - Communication
    - Ever play the operator game?
  - Documentation
  - Tooling
  - Hand-offs (process exchanges or role turn-over)

- Remember, you cannot read other people’s minds
Circle of Life

Teams come, operate, evolve or disband

People come, grow, and eventually move on

Projects come, grow, enter stasis or evolve
Project Influences

• Scale
  – Affects the ability to know “everything”
  – Complexity becomes a critical factor, if it wasn’t already

• Legacy
  – Rarely is everything from scratch
  – Being able to extend others’ work is essential
Professionalism

Personal Ethics

1. **Confidentiality**: Respecting confidences of employers or clients regardless if there is a formal agreement
2. **Competence**: Accurately reflect what you can do and accept only work that is within your competence
3. **Intellectual Property**: Protecting the IP of employers and clients
4. **Misuse**: Do not use skills or resources inappropriately

Effects

1. Developers and administrators may have access to highly confidential information
2. Systems that do not work can destroy a company
3. IPR violations can be result in fines or cease and desist orders
4. System abuse can paralyze a company

http://www.ieee.org/about/corporate/governance/p7-8.html

https://www.acm.org/about/code-of-ethics
Process

**noun process**

prä-, ses (US), prō-, -səs (CA)
– a series of actions that produce something or that lead to a particular result
– a series of changes that happen naturally

http://www.merriam-webster.com/dictionary/process

**Typical “Good” Qualities**

<table>
<thead>
<tr>
<th>Effective</th>
<th>Actually Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efficient</td>
<td>Reusable</td>
</tr>
<tr>
<td>Relevant</td>
<td>Managed</td>
</tr>
<tr>
<td>Valid</td>
<td>Measurable</td>
</tr>
<tr>
<td>Usable</td>
<td></td>
</tr>
</tbody>
</table>

! But it is easy to become over-zealous or lost in process
Software Process Models
aka Software Development Life Cycle (SDLC)

Purpose
- Lead to good software
- Reduce risk
- Enable visibility and measurement
- Enable teaming

At the highest level, there are four key elements

0. Feasibility
1. Specification
2. Architecture and Design
3. Development
4. Validation
5. Evolution/Maintenance

Key attributes of a process

1. Outcomes/results of processes are key deliverables or products
2. Roles are clear
3. Pre and post conditions are understood and held true

The devil is in the details of how the steps are organized and executed
The Promise of Good Process
The Reality Once You Get Downstream Some

Nothing is ever as simple as it seems
Sequential process phases
  – One step completes before next one starts

Rational process
  – Enables careful planning
    • This is how construction is done.
  – Good for
    • some piece of the system cannot be easily changed (e.g. hardware)
    • where explicit and exhaustive testing is required before launch

Challenges
  – Heavyweight process
    • Meaning the process is followed systematically and completely (slow)
    • Specification is a negotiation process
    • Specifications precede the system
  – World rarely is known upfront and even more rarely stays fixed
    • Hard to adapt to upstream changes once the step completes
Iterative Models

- System is created by successive versions.
  - Go through each process step, then iterate
    - Similar to how you are taught to write a paper
  - Includes feedback between steps
- Lowers the cost of implementing requirement changes
- Allows some client/user feedback to be considered
- Smaller sized steps mean delivery of something comes sooner
  - Value is created earlier
- It may not be clear where in the program the project is
- Changes can lead to messy designs and implementations

Iterative Models Diagram:

- **Feasibility**
  - Feasibility Analysis
- **Specification**
  - Requirement Document
- **Architecture & Design**
  - Design Documents
- **Development**
  - Code
- **Validation**
  - Test plan and results
- **Evolution & Maintenance**
Incremental Models (aka Agile)

• Variant on iterative models
• Emphasis
  – producing small increments of software in a reasonably short time frame called a sprint
  – Entire process is run during a sprint
  – Sprint results are deployed
• Antithesis of Waterfall
  – Very lightweight process
    • Plans develop incrementally
      – Client collaboration versus client negotiation
      – Specification follows from working system, not the reverse
    • Immediate feedback from deployment
      – Responding to change rather than following a plan
    • Enhancements, new features, and bug fix are all prioritized as candidates for focus during next sprint
    • Emphasis on keeping scope small
      – Although the impact of changes will grow over time

“[…] is like driving at night in the fog. You can only see as far as your headlights, but you can make the whole trip that way.”
— E.L. Doctorow, Writers At Work: The Paris Review Interviews
Test Based Design

• Puts test specification as the critical design activity
  – Understands that deployment comes when the system passes testing
Key Concerns Driving In Selecting a Process

- What is the net tolerance for finding errors in deployment?
- How fast is the market moving?
- Are the teams experienced with a particular process?
- Is the contract fixed and firm?
- When do the clients expect to see something?

Adapted from http://www.agilemodeling.com/essays/costOfChange.htm
Copyright © 2003 Scott W. Ambler
Even With Advances In Process Organizations, Project Success Remains Risky

**Pessimist View**

- All projects

**Optimist View**

- Lean
- Iterative
- Agile
- Ad-Hoc
- Traditional

Standish Group (UK), *Chaos Study*, 1995

Dr. Dobb’s Journal 2013 IT Project Success Survey posted at [www.ambysoft.com/surveys/](http://www.ambysoft.com/surveys/)
Feasibility

Determines if a project should be attempted

Feasibility study is a proposal

The decision maker is the audience

This person may not be sufficiently technical
Budget processes and staffing usually follow from a positive response

Three outcomes

1. Yea
2. Nay
3. nice try, think some more
What Goes into a Feasibility Study

Recommendation

Technology

Schedule

Economic

Operational

Legal
Uncertainty Makes This VERY Hard

Challenges

- Clients are unsure what they need at a useful level of detail
- Benefits are hard to quantify
- Impacts and recognizing unintended consequences is even harder to quantify
- Approach is often based on very rough guesses
- Organizational structures may need change
- Assumptions may be faulty

Mitigations

- Experience can guide process
  - But the most experienced people may not be the most technically current
- Solicit support and build interest for the project
  - Beware of irrational enthusiasm
    - Leads to unreasonable expectations
The Main Line

Senior member(s) of the client’s organization decide whether to begin a major software project.

Client: who is this project for?

Scope: is it well defined? Where are there dependencies and on whom?

Benefits: are the benefits real and quantifiable? Do I trust these numbers?

Technical: Is the project possible? Is there at least one technical way to carry out the project?

Resources: what are the estimates of staff, time, equipment, etc.?

What are the options if the project is not done?

Additional Considerations

- Do I trust this team?
- Have we tried this before?
- Market maker? Fast follower?
- Is this really worth investing in?
- Are there IPR issues?
- License dependencies?
- Can this organization pull this off?
  - Management capabilities
  - Development capabilities
  - Operational capabilities
  - Sales capabilities
A Report That Is Not Read And Understood Is Useless

You should take care to insure the document is well-written and all illustrations are coherent and reasonably well-drawn. After all, this is a presentation to the BOSS.

Assume a general audience, not just a technical audience

Make it as long as it needs to be, but NO LONGER.

Skip unnecessary details.

Every detail should be important for the executive to make a decision, not understand the domain.

For an example, see http://www.cs.cornell.edu/Courses/CS5150/2015fa/Feasibility1.pdf

(again, thanks to Bill Arms)