View of SE in this Course

- The **purpose of software engineering** is to **gain and maintain** intellectual and managerial control over the products and processes of software development.
  - **Intellectual control** means that we are able make rational choices based on an understanding of the downstream effects of those choices to deliver a system with the desired capabilities.
  - **Managerial control** means we are able to make rational choices about development **resources** to deliver a system on time and within budget.
- Begin to consider what this means in practice.

Need to Organize the Work

- Nature of a software project
  - Software development produces a set of interlocking, interdependent work products
    - E.g. Requirements -> Design -> Code
  - Implies dependencies between tasks
  - Implies dependencies between people
- Must organize the work such that:
  - Every task gets done
  - Tasks get done in the right order
  - Tasks are done by the right people
  - The product has the desired qualities
  - The end product is produced on time

Addressed by Software Processes

- Developed as a tool for controlling complex software developments (**separation of concerns)**
- Answers the “who”, “what”, “when”, etc. questions
  - What product should we work on next?
  - What kind of person should do the work?
  - What information is needed to do the work?
  - When is the work finished?
- Intended use
  1. **Model of development** (what does or should occur)
  2. **Guide** to developers in what to produce and when to produce it
    - Basis for planning and assessing development progress
Definitions

- **Software Life Cycle**: evolution of a software development effort from concept to retirement
- **Software Process Model**: Abstract representation of a software life cycle as a set of
  1. Activities or tasks
  2. Artifacts (work products)
  3. Roles
- **Organizational Software Process**: institutionalized version of a life software model defining specific roles, activities, and artifacts

A Simple Life-Cycle Model

<table>
<thead>
<tr>
<th>Problem</th>
<th>Requirements Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req Spec</td>
<td>Design</td>
</tr>
<tr>
<td>Design</td>
<td>Implementation</td>
</tr>
<tr>
<td>System</td>
<td>Testing</td>
</tr>
<tr>
<td>Working System</td>
<td>Maintenance</td>
</tr>
</tbody>
</table>

From van Vliet

Common Process Models

- Waterfall
- Prototyping
- Iterative
- Spiral
- Agile

A “Waterfall” Model
Activities and Products

- **Requirements Analysis**
  - Goal: implementation-independent specification of what the software must do and any constraints on its development
  - Product: Software Requirements Specification (SRS)
- **Architectural Design**
  - Goal: decomposition of the problem into components that together satisfy the requirements within the constraints
  - Products: specifications of components, relations, interfaces
- **Detail Design**
  - Goal: internal design of components (e.g., objects) to identify appropriate algorithms and data structures supporting the interface
  - Products: design documentation, pseudo-code

Phases and Products

- **Implementation**
  - Goal: realization of the design in a machine-executable language
  - Product: code
- **Testing**
  - Goal: validation and verification of the implementation against requirements and design
  - Products: test plan, test cases
- **Maintenance**
  - Goal: maintain deployed system
  - Products: bug fixes, patches, new versions

A “Waterfall” Model

Organizes by distinct software development concerns

What are the issues:
1. As a guide to how software should be developed?
2. As a model of any real development?

A “Waterfall” Model*

1. As a guide: does not address some common development risks
   - What happens if requirements are wrong?
   - Is scheduling or budget is wrong?
2. As a model: unrealistic as a model of any real development
   - How do real developments differ?

*not in text
Waterfall Model Variations

There have been many variations attempting to address these issues

Characteristic Model: Prototyping

- Waterfall variation
- First system versions are prototypes, either:
  - Interface
  - Functional
- Which waterfall risks does this try to address?

Characteristic Processes: The Iterative Model

- Process viewed as a sequence of iterations
  - Essentially, a sequence of waterfalls
  - Each iteration adds produces an increment of the working software

Iterative Model

- Also called “incremental development”
- Addresses some common waterfall risks
  - Risk that software cannot be completed – build incremental subsets
  - Risk of building the wrong system – stakeholder have opportunities to see the software
  - Also, feasibility, schedule, budget and others to some extent
Characteristic Processes: The Spiral Model

- Process viewed as repeating cycles of increasing scale
- Identify risks and determine (next set of) requirements, build next version by extension, increasing scale each time

Spiral Model Goals

- Response lack of risk analysis and risk mitigation in “waterfall” process
  - Make risk analysis standard part of process
  - Address risk issues early and often
- Explicit risk analysis at each phase
- Framework for explicit risk-mitigation strategies
  - E.g., prototyping
- Explicit Go/No-Go decision points in process

Characteristic Processes: Agile (scrum)

- Process viewed as nested sequence of builds (sprints)
  - Each build adds small feature set
Also…

- RAD models
- Extreme Programming
- Etc., etc.

Why so many models?

Take-away

- Expected to know standard processes and their rationale
- Helps organize understanding to tasks in a software development
- For Projects: what kind of process should you use?
  - Understand how and why people use different development models
  - Understand how to choose an appropriate model for your developments
  - Understand how to map processes to plans

How do we Choose a Development Process?

E.g., for your projects

Objectives

- Goal: proceed as rationally and systematically as possible (i.e., in a controlled manner) from a statement of goals to a design that demonstrably meets those goals within design and management constraints
  - Understand that any process description is an abstraction
  - Always must compensate for deviation from the ideal (e.g., by iteration)
A Software Engineering Perspective

- Question of control vs. cost
- Choose processes, methods, notations, etc. to provide an appropriate level of control for the given product and context
  - Sufficient control to achieve results
  - No more than necessary to contain cost and effort
- Provides a basis for choosing or evaluating processes, methods, etc.
  - Does it achieve our objectives at reasonable cost?
  - E.g., does this notation provide a handle on the properties of interest?

Project Relevance

- Need to agree on kind of control you need and how you will accomplish it
- Process model will then help keep everyone on track
  - Basis for planning and scheduling
  - Each person knows what to do next
  - Basis for tracking progress against schedule
- Should be one of the first products you produce but expect it to evolve

Contents of a Process Specification

- Details depend on the purpose of the specification
- In general terms [Parnas &Clements]
  - What product we should work on next?
    - Equivalently – what decision(s) must we make next?
  - What kind of person should do the work?
  - What information is needed to do the work?
  - When is the work finished?
  - What criteria the work product must satisfy?
- In personal terms, answers the questions
  - Is this my job?
  - What do I do next?
  - What do I need to do the work?
  - Am I done yet?
  - Did I do a good job?

Project Processes

- Discuss: what process elements are appropriate for your project?
- What are the products?
- What aspects of traditional models are irrelevant?
- What are the constraints?
  - Which aspects can't be changed?
  - Which can be?
- What are the major risks?
- What are appropriate strategies to address the risks?
Questions?