These are Anthony Hornof’s notes from:


Merriam Webster and New Oxford American dictionaries.

The notes were taken to (a) learn and organize an understanding of the material and (b) prepare lectures. The notes are not at all complete. All chapters and sections are not included here. You must do the reading itself to learn the required course material.

Some of the notes are copied directly from the textbooks.

Part 1 in "Contents at a Glance" provides a good overview of most of the assigned reading.

The content in boxes, such as this, is *not* from the book.

These notes are organized around the chapters in Sommerville (2015).

**Table of Contents**

<table>
<thead>
<tr>
<th>Chapter 1 - Introduction</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 5 - System Modeling</td>
<td>5</td>
</tr>
<tr>
<td>Chapter 22 - Project Management</td>
<td>10</td>
</tr>
<tr>
<td>Section 23.3 - Project Scheduling</td>
<td>16</td>
</tr>
</tbody>
</table>
Chapter 1 - Introduction

Four themes that pervade all aspects of software engineering. (A. Hornof)

1. **Use abstractions.** Find ways to summarize detailed specifications of concepts and ideas, and use these brief summaries in place of the more complex ideas. It is imperative that all team members understand the abstractions that other team members are using.

2. **Divide and conquer.** Break a large problem into smaller pieces that can be solved individually.

3. **Propose and consider alternatives.** Nearly every activity in software engineering involves some form of design, in which design is the process of proposing and evaluating alternative solutions to a problem.

4. **Collaborate.** Most of the processes and techniques developed and used in software engineering are ultimately aimed at assisting groups of two or more people combine their brainpower to solve a problems together.


Read the first page in class.

1.0 - **Intro to Chapter 1.**

Software engineering is essential for the functioning of government, society, and national and international businesses and institutions....

Software systems are abstract and intangible. They are not constrained by the properties of materials, nor are they governed by physical laws or by manufacturing processes....

There are many different types of software system, ranging from simple embedded systems to complex, worldwide information systems....
There are still many reports of software projects going wrong and of “software failures.”

1. Increasing system complexity.

2. Failure to use software engineering methods.

**What is software?**

Computer programs and associated documentation, libraries, support websites, and configuration data that are needed to make these programs useful.

**What is software engineering?**

Software engineering is an engineering discipline that is concerned with all aspects of software production from initial conception to operation and maintenance.

**What is the difference between software engineering and computer science?**

Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.

If you are writing a program for yourself, no one else will use it and you don’t have to worry about writing program guides, documenting the program design, and so on. However, if you are writing software that other people will use and other engineers will change, then you usually have to provide additional information as well as the code of the program.

**1.1.1 Software engineering**

Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use. In this definition, there are two key phrases:

1. **Engineering discipline.** Engineers make things work. They apply theories, methods, and tools where these are appropriate, and they must look for solutions within constraints.
2. **All aspects of software production.** Software engineering is not just concerned with the technical processes of software development. It also includes activities such as software project management....

Software engineers adopt systematic and organized approaches to their work, but engineering also requires a creative approach to solving problems.

[Skipping 1.1.2 through the end of the chapter. Students read on their own.]

[Note that 1.3 introduces the case studies used throughout the book.]

---

**What is Software Engineering?** (from Stuart Faulk)

Software engineering is the process of gaining and maintaining control over the products and processes of software development. There are two kinds of control:

- “Intellectual control” means that we make rational choices based on an understanding of the effects of those choices on the qualities of the product and process.

  *Such as understanding the implications of using C++ versus python.*

- “Managerial control” is related but different in focus: The purpose is to gain and maintain control of software development resources (money, time, personnel).

  *Such as figuring out whether to try to hire more programmers or delay the delivery date.*

In practice, both are necessary and inseparable. It would difficult to have managerial control if you do not first have intellectual control.

In contrast to computer science (the broad study of the basis and behavior of computing machines), software engineering is an inherently pragmatic discipline.
Chapter 5 - System Modeling

The chapters introduce a number of diagramming techniques that are commonly used to communicate aspects of a system design. The diagrams are called “models” because they serve as small-scale representations, or paper-based simulations, of aspects of the system.

‘The fundamental driver behind graphical modeling languages is that programming languages are not at a high enough level of abstraction to facilitate discussions about design.’ (Fowler, 2004.)

The models are static or dynamic. Static show structure. Dynamic show behavior.

Flowcharts are a classic dynamic model to show the flow of control of an algorithm. UML activity diagrams are very similar. UML uses the terms "flow" and "edge" synonymously. (Fowler)

In any diagram, you generally need a key that explains what the boxes and lines represent. However, if you are correctly using an established diagramming technique, citing a source can suffice.

A simple activity diagram, annotated (Fowler, 2004)
The Unified Modeling Language
Diagramming techniques used in OOA and OOD (analysis and design). Integrates and “unifies” the notations and methods of Booch, Jacobson, and Rumbaugh (object modeling technique, OMT), late 1980s and early 1990s. There is also a UML process, but the language is still useful without the process.

(UML notes adapted from Sommerville, 2000, Software Engineering.)

There are other standard diagramming (modeling) techniques such as:
1. Entity Relationship Diagrams (ERDs) - similar to UML class diagrams.
2. Data Flow Diagrams - similar to UML sequence diagrams.
This lecture focuses on UML.

There are 13 different UML Diagrams, in the following hierarchy:

Structure:  
- Class
  - Component
  - Composite Structure
  - Deployment
  - Object
  - Package

Behavior:  
- Activity
- Use Case
- State Machine

Interaction:  
- Sequence
  - Communication
  - Interaction Overview
  - Timing

Boxes and lines mean different things in each type of model.
Note how there is a fundamental distinction between static and dynamic.
Major diagrams used in UML

Class diagrams: Static. Descriptions of the types of objects in the system, and the various kinds of static relationships that exist among them.

State-transition diagrams: Dynamic. Show all possible states (modes) that an object can get into as a result of events that reach that object.

Sequence diagrams: Dynamic. Describe how groups of objects collaborate in some behavior. Show the sequence of object interactions

(UML notes adapted from Sommerville, 2000, Software Engineering.)

UML Class diagrams

Descriptions of the types of objects in the system, and the various kinds of static relationships that exist among them. Static model.

Include: Name of class, attributes and operations, inheritance (specialization). Associations, such as is-a-member-of, cardinalities.

A simple class diagram, annotated (adapted from Fowler, 2004, Figure 3.1)
But *dynamic* models are also necessary to describe how a computer program works because a program executes over time. (A screenshot does not describe a user interface; you also need to describe the dynamic aspects.)

**UML State Diagrams**
A *dynamic* model illustrates the restricted states of an object or system. The ovals are states and the arcs are events that cause the state to change. Can have hierarchies of states, introducing abstraction. Permit stakeholders to understand the of dynamic aspects of the system.

![State Diagram](image)

A state diagram for a weather station that, every five minutes, collects data, performs some data processing, and transmits this data. (From Sommerville, 2000, Software Engineering)

**UML Sequence Diagrams**
Dynamic models that describe how groups of objects collaborate to produce a system service or behavior. Shows the sequence of object interactions. Objects and users are shown at the top, each with vertical dashed *lifelines*. Rectangles on the lifelines show when the object is active. Time moves down. Solid lines show messages between objects. Dashed lines indicate a return.
The difference between a State Diagram and a Sequence Diagram
A state diagram says “All allowable sequences must conform to this state machine” whereas an interaction diagram says “Here is one possible sequence of actions.” (Prof. Young, 11-9-2010)

Conclusion: UML evolved from earlier OOA and OOD methods, which evolved from earlier non-OO diagraming and design techniques. All diagramming (modeling) techniques arrive at roughly the same models. When you think about a piece of code that you are going to write, you think about the static and dynamic aspects of how that code will work. Use standardized diagramming techniques to sketch out your ideas, both for yourself to think things through, and to communicate, record, and evaluate ideas with other team members and stakeholders. Software design modeling is an important aspect of software engineering, the study of the full lifecycle of writing the code that run on computers.
Chapter 22 - Project Management

These notes are primarily copied from Sommerville (2015), and cover just a subsection of the assigned reading. Students should do all of the reading on their own following the SQ3R method.

Section 22.2 Managing People

Productivity is achieved when people are respected by the organization and are assigned responsibilities that reflect their skills and experience.

Four critical factors that influence the relationship between a manager and the people that he or she manages:

1. Consistency. People are treated the same, and held to the same expectations (given each individual's ability to contribute).

2. Respect. Different people have different skills. Everyone should be given the chance to contribute. [In this class, each student should be given a good opportunity to make a technical contribution.]

3. Inclusion. All ideas from all team members should be considered. Try to develop participation techniques to elicit contributions from team members who are more reflective, and less assertive in meetings. [Such as a quiet individual brainstorm followed by input from everyone.]

4. Honesty. Everyone should be clear and up front about what is going well, and what is not going well.

"The only thing worse than bad news is bad news late."

Section 22.2.1 Motivating People

One way to think about motivating people is in the context of Maslow's hierarchy of human needs. In this hierarchy, each lower needs must be met before any of the higher needs can be met.
Figure X. Maslow's Hierarchy of Human Needs
(from https://www.simplypsychology.org/maslow-pyramid.jpg - 1-10-2022)

Human needs, starting from the bottom of the hierarchy.

1. **Physiological.** Team members must get enough sleep have adequate access to food. [Randy Pausch's Tips for Working Successfully in a Group.]

2. **Safety.** This includes team members feeling completely unthreatened in the workplace. ["Sexual Harassment In Silicon Valley: Still Rampant As Ever". September 15, 2020. *Forbes.*]

3. **Belongingness.** Team members should be recognized and appreciated as individuals. ["I see you." "I hear you."]

4. **Self-esteem.** People's contributions to the project, and to meetings, should be acknowledged.

5. **Self-realization.** People should be able to work at their level of ability, and to learn new things. [This does not mean to "follow your dreams".]
Maslow's hierarchy is a useful framework, but it does take a somewhat self-centered perspective, which can conflict with the need for a group to be cohesive and work well together. [Ask instead how you can contribute.]

**Project Questions on Motivating People:**

What are some ways that your group could better address either (a) Maslow's hierarchy of human needs or (b) Randy Pausch's "Tips for Working Successfully in a Group"?

Devise a specific proposal or request, and describe exactly how you will present it to your group.

**Section 22.3 Teamwork**

Teams need to be managed.
This is a task unto itself.
It requires consideration of alternatives.

*cohesion* means "sticking together tightly" or "forming a united whole".

A cohesive group values the group more highly than individuals in the group. Members of a well-led cohesive group are loyal to the group.

**Benefits to a cohesive group include:**

1. When the group makes decisions independently of outside influences, this contributes to a sense of independence and autonomy, and also of belonging.

2. Team members learn from each other.

3. Knowledge is shared so that people can help cover each other's tasks.

4. There is continual improvement to the overall product, not just parts of it.

Good project managers encourage group cohesiveness.
"One of the most effective ways of promoting cohesion is to be inclusive." Treat group members as responsible and trustworthy, and make information freely available to everyone in the group. Everyone should know what is going on, should be able to name and contact all other group members, and have in mind at all times a general idea of what everyone is working on.

**Project Questions on Teamwork:**

What are some activities that your group does to promote cohesion? Inclusion?

Does everyone in the group know everyone else in the group, and have one or more ways to communicate with that person? Such that the whole group can see the communication?

Does everyone in the group know what everyone else is working on? If not, what are a few different ways that could be improved?

What are some ways that any of the above might be improved?

**Three factors that have a big effect on team work include:**

1. **Who is in the group.** There should be a mix of skills.

2. **How the group is organized.** People should be able to contribute at their level of ability, and complete tasks as expected.

3. **Technical and management communication.** Good communication among all team members is essential

**22.2.3 Group Organization**

Important organizational decisions include:

1. Should the project manager and technical lead be the same person?
2. Who will be involved in making critical decisions, and how will the decisions be made?
3. How will interactions with external stakeholders be managed?
4. How will groups interact with team members who are not co-located?
5. How will knowledge be shared across the group?

**Project Questions on Group Organization:**

- How are decisions being made about who will do what?
- How are technical decisions being made?
- Are there any policies about how to contact the professor?
- How are co-located team members included (during Covid)?
- What are some ways that any of the above might be improved?
**Project Questions on Motivating People:**

What are some ways that your group could better address either (a) Maslow's hierarchy of human needs or (b) Randy Pausch's "Tips for Working Successfully in a Group"?

Devise a specific proposal or request, and describe exactly how you will present it to your group.

**Project Questions on Teamwork:**

What are some activities that your group does to promote cohesion? inclusion?

Does everyone in the group know everyone else in the group, and have one or more ways to communicate with that person? Such that the whole group can see the communication?

Does everyone in the group know what everyone else is working on? If not, what are a few different ways that could be improved?

What are some ways that any of the above might be improved?

**Project Questions on Group Organization:**

How are decisions being made about who will do what?

How are technical decisions being made?

Are there any policies about how to contact the professor?

How are co-located team members included (during Covid)?

What are some ways that any of the above might be improved?
Section 23.3 - Project Scheduling
Some of these notes are from material that is not in Sommerville (2015).

Recall that software engineering is the process of gaining and maintaining control over the products and processes of software development.

• “Intellectual control”...
• “Managerial control” focuses on gaining and maintain control over software development resources (money, time, personnel).

This lecture focuses on control of the resources of time and personnel. 

*Plans are nothing. Planning is everything.* (Attributed to President Eisenhower)

“Begin with the end in mind”. (Franklin Covey. 1989. The 7 Habits of Highly Effective People)

PERT Charts
Process Evaluation and Review Technique
(Developed during the 1950s Polaris missile program.)
The basic idea: Each activity gets a box. Lines indicate the necessary completion order (because of some kind of constraint).

```
<table>
<thead>
<tr>
<th>Activity</th>
<th>Duration</th>
<th>Necessary completion order (from left to right)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan castle</td>
<td>5 days</td>
<td></td>
</tr>
<tr>
<td>Dig moat</td>
<td>7 days</td>
<td></td>
</tr>
<tr>
<td>Build walls</td>
<td>20 days</td>
<td></td>
</tr>
<tr>
<td>Build tower</td>
<td>20 days</td>
<td></td>
</tr>
<tr>
<td>Move in</td>
<td>3 days</td>
<td></td>
</tr>
</tbody>
</table>

Key:
- Activity duration
- Necessary completion order (from left to right)
- Critical path

PERT charts emphasize the *critical path*....
Project Planning Terminology

The **critical path** is the sequence of activities in a project such that, if any of these activities is delayed, the entire project is delayed.

You should always be working on activities that are on the critical path.

In the example above: How many days to complete project? What happens if the steam shovel breaks you have to dig the moat by hand, for 50 days?

**Slippage** is the time a task (or project) is late compared to the original deadline. Slippage only delays the project if it is on the critical path.

**Slack time** is the time that a task can be delayed without delaying the project.

**Milestones** are distinctly identifiable points in the project timeline, named after stones that appear along the side of a road.

**Deliverables** are well-defined physical or digital objects that are handed over from one stakeholder to another. Every deliverable can be a milestone, but every milestone does not necessarily have a deliverable associated with it (such as simply starting a task).

Gantt Charts (Timelines)

Named after Henry Gantt. (He developed them around 1910 to maximize the productivity of factory workers.)

The basic idea is as follows, though they can be drawn in many different ways. Time always moves from left to right.

The time scale depends on the size of the project and the scope of the chart.

<table>
<thead>
<tr>
<th>Task</th>
<th>Staff</th>
<th>Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan Castle</td>
<td>QN</td>
<td>5</td>
</tr>
<tr>
<td>Dig Moat</td>
<td>KG</td>
<td>7</td>
</tr>
<tr>
<td>Build Walls</td>
<td>PC</td>
<td>20</td>
</tr>
<tr>
<td>Build Tower</td>
<td>QN, KG</td>
<td>20</td>
</tr>
<tr>
<td>Move In</td>
<td>All</td>
<td>3</td>
</tr>
</tbody>
</table>

The critical path can be drawn, but it is not as visually emphasized as it is in the PERT chart.

Other columns can be added, such as start and end dates, resources needed, etc. Gantt charts emphasize task duration, start/end dates, and task overlap.
Both diagrams are very useful but can be tedious to keep up-to-date. Both are extremely useful for planning and communicating. They must be updated regularly. Save a dated copy every time you update.

I recommend using a spreadsheet or direct drawing editor, not a complicated task management software such as Asana. You want easy and direct editing on a single page.

The diagrams provide well-established conventions. If you can communicate time and task needs, you can gain power and control. (LTCB story: 1 week, 2 weeks, 3 weeks, 2 weeks.)

**Project Questions on Project Scheduling:**

Draw a Gantt chart to explain to a student in a different group (and then to the class) how your group is using its time. This can be explained at the level of the entire group, or at the level of each individual team member.

Propose two fundamentally different order of activities that could be done on this project. Each of the two should have real strengths. Start with a brainstorm of all of the activities that need to get done. Generate a different graphical view of each order of activities.

Each Gantt chart should include indications of the critical path.