1 Reading 1: HFSD Chapter 1, Page 1-26

- If your customer is unhappy, then everyone is unhappy.
  - Every great piece of software starts with a customer's big idea.
  - As a software developer, you bring that idea to life.
  - A vague idea to working code – one that satisfies your customer – isn't easy.
  - You need to learn how to deliver software that is **needed**, **on time**, and **on budget**.
- Most projects have two major concerns.
  - How much will it cost? Most customers want to figure out how much money they'll have to spend. Money may be a limitation.
  - How long will it take? Almost no customer ever says "Take as long as you want!" Also, in a lot of cases, there's a specific event or date (i.e., a deadline) the customer wants their software ready for.
- The Big Bang approach to development.
  - Work a lot, then **bang!** Something huge and *complex* comes out of the work all at once.
  - Known as "going dark," as the customer sees you at the beginning of the project, and then you disappear until the software is delivered at the end.
  - Usually ends up in a **big mess**.
    - ∗ The customer may not be happy with the resulting software.
    - ∗ What the customer had in mind may be very different from what you had in mind, and because you never talked to the customer, you wouldn't know what the customer wanted (aside from some rough ideas at the beginning when you talk to the customer).
    - ∗ If your customer isn't happy, then you built the wrong software.
      - · Essentially, big bang software usually means working a whole lot. It also means not showing the customer much until your work is done. The risk with that approach is that you think you're building what the customer wants with no real feedback until you think you're finished.
      - · No matter how great *you* think your software is, it's the customer you have to make happy.
      - · How do you know what the customer really wants?

> For example, in the reading, Tom wanted a website to showcase the "next evolution in hiking," as a website. Basically, take his business to the internet. The rough ideas he had in mind were
>
> - ∗ The customer should be able to search for trails.
> - ∗ The customer should be able to order equipment.
> - ∗ The customer should be able to book a trip.
>
> These are all very general.

- If you aren't sure what the customer wants, or even if you think you're sure, **always** go back and ask.
  - When it comes to what is needed, the customer is king. But, it's rare for the customer to know exactly what they want at the beginning of the project.
  - Software should not be guesswork. You need to ensure that you develop great software even when what's needed is not clear up front.
  - Ask!

    * Ask the customer what they mean.
    * Ask them for more details.
    * Ask them for options about how you might implement their big ideas.

> Software development is not guesswork. You need to keep the customer in the loop to make sure you're on the right path.

- What is great software development? Great sofware development delivers
  - what is needed (what the customer needs, otherwise called the **software requirements**),
  - on time (when we agreed with the customer that the software would be finished), and
  - on budget (not billing the customer for more money than was agreed upon).
- Getting to the goal with **iteration**.
  - The ideal development path is the one that the customer wants. That path will lead to your ultimate goal: a product that the customer wants.
  - Without iteration, you may or may not follow that development path, and may – as a result – miss the goal (i.e., give the customer a product they didn't want).
  - With iteration, you're constantly checking up with the customer about their requirements, adjusting the development path you're taking as needed. At the end, you should hit the goal.

> Iteration is like a frequent checkup for your software. You'll always know how you're doing.

  - Iteration (and getting feedback) from your customer is important especially when you think you know it all up front. You always want to make sure you're on the right track.
  - Iteration is useful even on a short project.
  - Getting to know what the customer really wants, and getting the requirements down at the very beginning, is still not ideal. The customer often doesn't understand what they want, or may change their mind. You still need to check with the customer from time to time.
  - Everyone who has a say in whether your software meets its requirements, and everyone who is involved in meeting those requirements, should be involved in iteration. Usually, this is you, your customer, and other developers working on the same project.
  - Iteration is still necessary even if you're a solo developer.
  - As soon as you have a piece of software running, discuss it with your customer. Recommended time is usually around 20 work days (1 calendar month, per iteration)[1].
- Iteration produces working software.
  - With iteration, you check every step of the way that you're going in the right direction.
  - This means making sure your software builds from almost day one. You shouldn't have long periods where code doesn't work or compile, even if it's just small bits of functionality. More productivity, as you don't need to spend time fixing someone else's code before getting on with your own tasks.

> With iteration in mind, we can show Tom small portions of the website (breaking up each task into smaller chunks of functionality). Tom gets to see working software and can make some important comments that you can address right away.

- Each iteration is a mini-project.

[1]This is usually a guideline

  - With iteration, you take the steps you'd follow to build the entire project, and put those steps into each iteration.
  - Each iteration is a mini-project, with its own requirement, design, coding, testing, etc. built right in.
  - Your customer basically gets to "sign off" on what you've built.

- Each iteration is quality software.
  - Think of iteration as little cycles, where you're gathering requirements, designing, writing code, and testing. Each cycle produces working, quality software.
  - An iteration contains all the stages of a complete process.
  - After multiple iterations, where each iteration builds a small piece of software, you should end up with the final software.

- Your iteration length should be at the right tempo for your project.
  - Remember, an iteration can be shorter or longer than 30 days. Try 30 calendar days first, and then tweak for project as needed.
  - The key is to iterate often enough to catch yourself when you're deviating from the goal, but not so often that you're spending all your time preparing for the end of an iteration.
  - It takes time to show the customer what you've done and then make course corrections, so make sure to factor this work in when you are deciding how long your iterations should be.
  - Some other things to remember.
    * If the last feature scheduled for an iteration pushes the time needed to over a month, consider moving that feature to the next iteration. Going longer runs the risk of getting off course.
    * Even if you order things by customer priority, if you have features that need to be completed before other features, try to group those features together. This can be done even if a lower-priority feature needs to be done before a high-priority feature (as long as it makes the high-priority feature possible).
    * Adding more people to a project *can* help you do more in each iteration, but doesn't necessarily *halve* the time it takes to complete a feature.

- The customer will change things up.
  - You need to make the adjustments.
  - But, what if you're a long way into development, you have other features to build, the deadline hasn't changed, but the goal has moved (e.g., new feature added)? Iteration soft of handles this automatically.
    1. Estimate new features. Estimate how long each of the new features will take.
    2. Have the customer prioritize the new features.
    3. Rework the iteration plan (remember that ordering is based on prioritization; you may need to push some stuff to a new iteration).
    4. Check the project deadline. Days of work left - Days left before deadline = Can you do it?

- A process is really just a sequence of steps.
  - A process, particularly in software development, has gotten a bit of a bad name. It's just a sequence of steps that you follow in order to do something, like develop software.
  - Iteration, prioritization, and estimation are part of the software development process.
  - Doesn't really matter what process you use, as long as it has the components that ensure you get great, quality softeare, at the end of your development cycle.

- Iteration is more than a process.

  - It's an approach that can be applied to any process.
  - Gives you a better chance of delivering what is needed, on time, and on budget.

- Your software isn't complete until it's been released.
  - Once you deliver your software, then you get paid.
  - Iteration helped you reach an achievable goal that captured what your customer needed.
  - Some things to remember.
    - When the customer comes up with new requirements and you can't fit all the extra work into your current iteration, the customer priority comes into play. Your customer needs to make a call as to what really needs to be done for this iteration of development.
    - If you're at the last iteration and a top priority feature comes in from the customer, you need to explain to the customer why the feature won't fit. Show them your iteration plan and explain why, with the resources you have, the work threatens your ability to deliver what they need by the due date. Factor the new requirement into another iteration on end of project, extending the due date if the customer agrees.