Lifecycle modeling

Tugberk Kocatekin
T.C. Istanbul Arel University
Spring 2023

Introduction

- There are different software process models.
- **Waterfall model**
  - Plan-driven. Separate and distinct phases of specification and development.
- **Incremental development**
  - Specification, development and validation are interleaved. Maybe plan-driven or **agile**.

Problems in Software Development

- Requirements are constantly changing.
  - It is possible that even the client might not know all the requirements in advance. Technology improves and needs change.
- Frequenct changes are difficult to manage
- New system must be compatible with the old system
  - We see the needs and develop a new software, but it must be backward compatible. Otherwise we cannot reach to other data.

Waterfall Model

- Here, each phase must be completed before the next phase can begin.
  - There is no overlapping in the phases.
- It is the easiest approach. But not popular anymore.
- Simply, you don't get to the next phase if you didn't finish your current phase.
- Compared to the systems before, Waterfall model is an improvement. However, is it enough in this day and age?

Waterfall Model

- Requirements
  - System and Software Design
    - Implementation and Unit Tetsing
      - Integration and system testing
        - Operation and maintenance

Waterfall

- Managers love this model because there are nice milestones.
- There is always one activity at a time.
  - Very easy to check progress.
    - 90% coded, 20% tested.

However

- Software development is non-linear.
- While designers are doing design work, it is possible that they can find problems in *requirements*.
- While developers are coding, they can find bugs in design ,etc.
- Therefore, these are all together. It is not logical to go step by step.

Agile models

- Before, software development was different. Companies were focused on excessive planning and documentation and forgot about customer satisfaction.
- Several people came together and created `Agile Manifesto`. You can read the Agile Manifest here.
- The term *agile* is not new, it was before the manifesto. It is a **general term** (philosophy).
- XP and Scrum are the most used techniques.

Agile

- Remember when we said Waterfall model is not enough for changing systems. Agile is better for those.
  - That is why, generally Agile is used in software development.
- Instead of going step by step, *specification*, *design*, *implementation* and *testing* are interleaved.
- Agile methods attempt to develop a system *incrementally*, by building a series of prototypes and constantly adjusting them to user requirements.
- Emphasizes continous feedback and each incremental step is affected by what was learned in the prior steps.

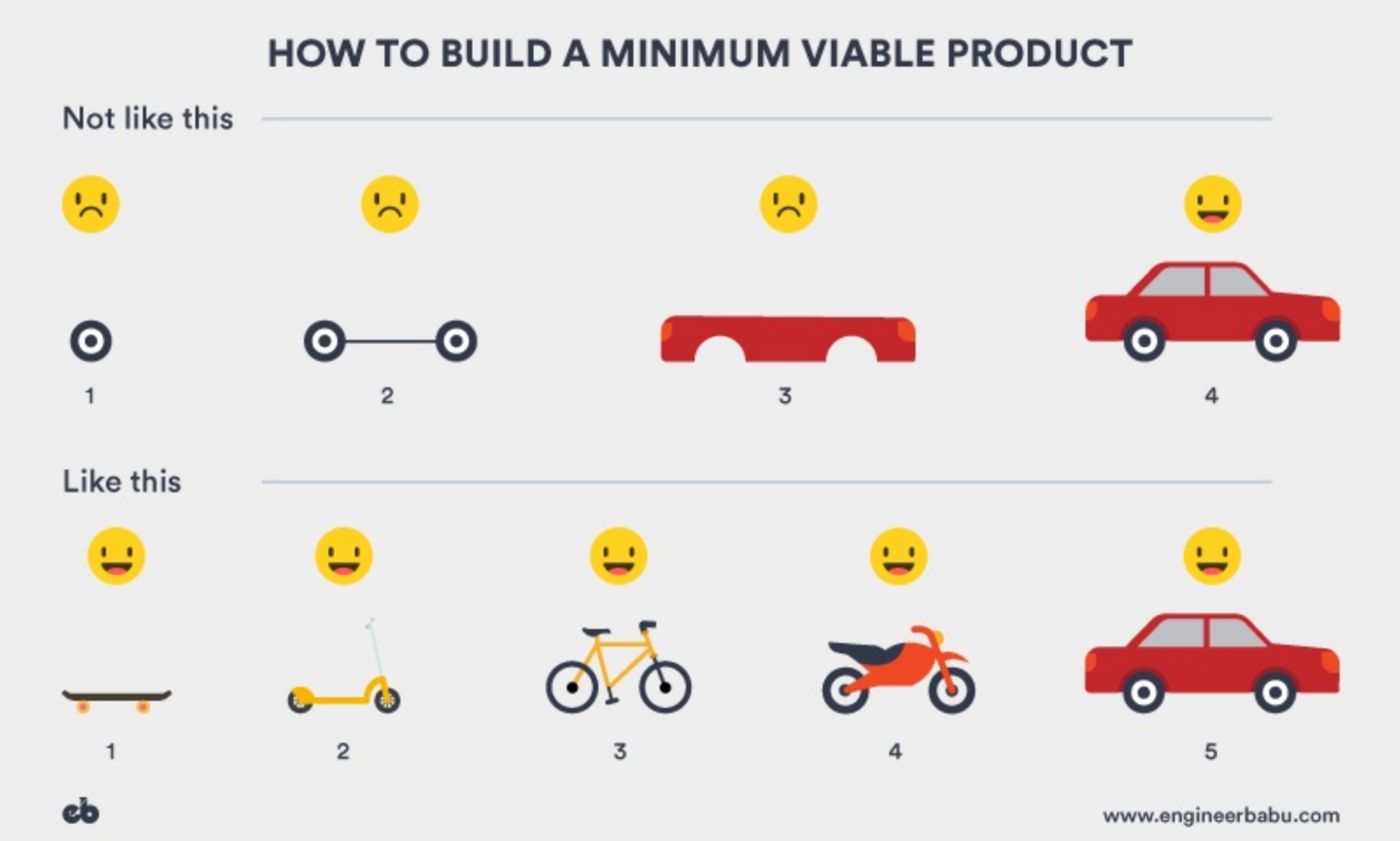
Principles of Agile Methods

- Customer involvement
  - Customers are closely involved in the development process. Interviews etc.
- Incremental Delivery
  - Software is developed in increments with customers specifying requirements in each increment. In every increment, a new feature or product is added to the system.
- Embrace change
  - System should be designed in a way that it can accommodate changes.
- Maintain simplicity
  - Try to eliminate complexity from the system.

MVP: Minimum Viable Product

- A version of the product with *just enough* features to be usable by early customers so that they can provide feedback.
- Most important thing in MVP is that it actually *does* the job.
- Every product you ship **should** do a part of the big picture. Every version should be working. It must be adapted to the end goal.

MVP


HOW TO BUILD A MINIMUM VIABLE PRODUCT
Not like this
1
2
3
4
Like this
1
2
3
4
5
www.engineerbabu.com

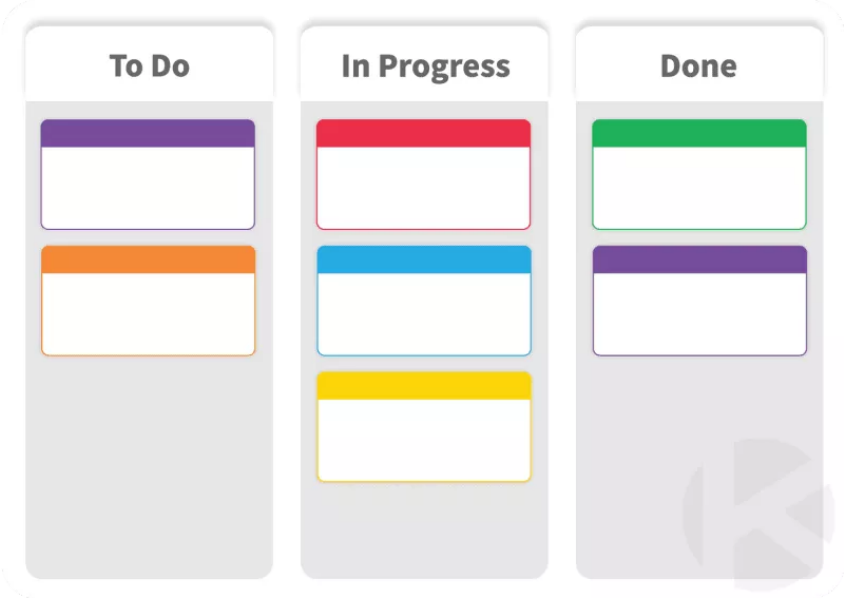
Problems

- Here, team members are involved in almost every process.
  - We will see in Scrum, there are constant meetings.
- Maintaining simplicity is not easy.
- Changes are sometimes hard because there are multiple stakeholders.

Kanban

- Before talking about Scrum and XP, let's just talk about Kanban a little bit.
- It is not only for software or system development, this approach can be applied to almost everything, even in daily life.
- Kanban is an **agile** project management tool.
  - It means *visual signal* in Japanese.
- It uses **kanban boards** which makes every work visible and everyone can see them, so everyone is on the same page.

Kanban board

- Although there are more sophisticated methods, let's start with the simplest one.
- What we need is some Post-it notes and a wall.
- We divide the wall into 3 sections: Todo, Doing and Done.
- 

- A physical example.

Kanban board

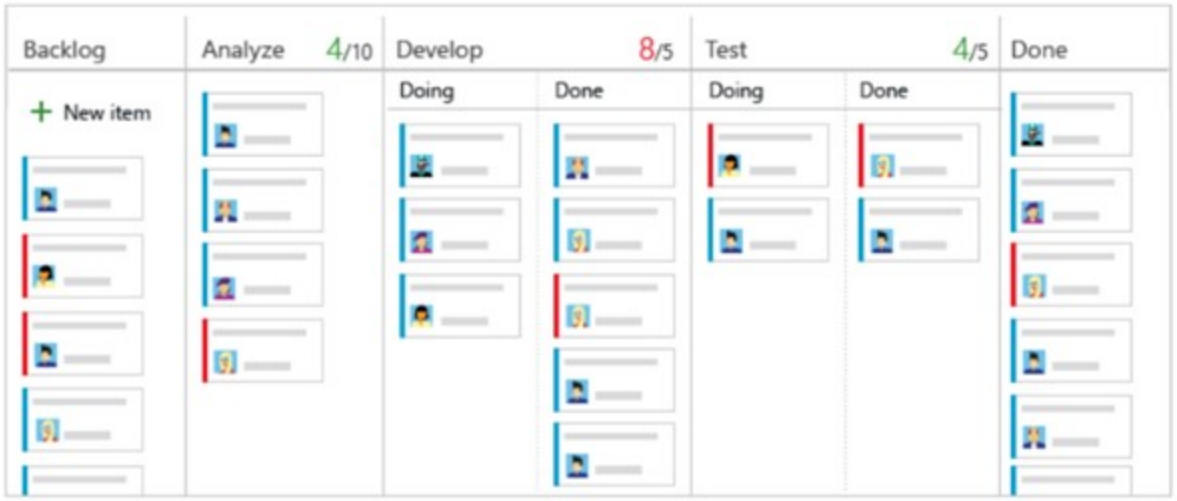
- Todo part is also called a **backlog**. Here, we write down every single task we need to do.
- When you want to do a task from the todo list, you get the post-it and stick it to the "In Progress" column. That means someone is doing that task!
- When done, you put it to Todo part.

Digital Kanban Board

- There are also digital kanban boards, which are the norm right now.
- Trello is a free web application where you can create Kanban boards for yourself. When you check the website, you can see there are a lot of use-cases!
  - I really recommend you take a look at it and start using it.
- In business, many companies use **Jira** which is an **Atlassian** product.

Advanced Kanban Board

- Another approach is dividing each column into two parts: Doing and Done. (those are called **swimlanes**)



- Here, every task has a separate **Done** section. For example here, Developers put the task into Done part when they coded it. Test team can now take tasks from the **Done** of Develop board to their **Doing** section.

Scrum

- It is an **agile** project management framework.
- Almost all companies are applying Scrum.
- Scrum works by **sprints**.
  - For further information on sprints, check here.
- Sprint is a time-boxed period (usually 2 weeks) when a team works to complete a **set** amount of work.
  - It is the heart of agile and scrum methodologies.

Sprints

- Sprint is a serious work. Before every sprint, you do a **sprint planning** meeting.
  - It is a collaborative event where the team answers two basic questions:
  - What work can be done in this sprint?
  - How will it get done?
- For every sprint, we must choose the **right** work.
  - We choose tasks from the **product backlog**.

Product backlog

- Remember the Todo?
  - Here we have product backlog. However, there are differences.
- In Kanban, we don't have **sprints**. Developers can take whatever they want.
  - Although it is useful, it is not widely used for software development. It is generally better for **bug fixes** or **devops**. It is a very good project management tool.

Sprints

- Before every sprint, teams do a sprint planning meeting and decide which items from the **product backlog** will be in this **sprint**.
- Choosing the right work for every sprint is important and not an easy task. For a sprint to be successful, the team must do every task there is successfully.
    - The team must be used efficiently. We must select the right difficulty and number of tasks for every sprint.

Daily Scrum (standup)

- Every morning, a daily meeting is done with the team to discuss the progress and identify blockers. Product owners and developers come together.
  - A blocker is a task where it prevents you working.
- It should be kept short because it is not about **planning**. It should be done in Spring Planning.
- Also called *huddle*.
- Here are some basic questions:
  - What did I work on yesterday?
  - What am I working on today?
  - What issues are blocking me?

Sprint Review

- Done after a sprint.
- In this meeting, stakeholders are also there in addition to the team. Here, teams showcase their work to stakeholders and teammate before production.

Spring Retrospective (retro)

- Purpose of *retro* is to plan ways to increase quality and effectiveness.
- This **concludes** the sprint. It is done at the end of the sprint.
  - It should be brief. It should not take hours.
- Team discusses following stuff:
  - What went well in the sprint?
  - What could be improved?
  - What will we commit to improve in the next sprint?
    - Be careful, it is **not** about planning the next sprint.
- By the end, scrum team should have identified improvements that they will implement in the next sprint.

Basic Example

- I have a backlog of my own. Creating notes, exams and homeworks for every class there is.
- All of these are written to the Backlog.
- Kanban or Scrum-like problem?
  - Not Kanban. I need to prioritize my work.
- Let's create a Kanban board as an example for this.

Extreme Programming (XP)

- Part of *agile*.
- The aim is to remove the resistance to changing code.
  - You must *embrace* change.
- Works best if:
  - under 10 people because it is *highly* collaborative.
  - are in constant contact with customers
  - developers are beginners.
- For further information see: here

Other Development Methods

- **JAD (Joint Application Development)**
  - Focuses on team-based fact-finding
  - Users, managers and analysts work together for several days.
  - System req. and designs are reviewed.
  - Intensive and structured meetings.
- **RAD (Rapid Application Development)**
  - A compressed version of the entire development process.
  - Follows the same phases in SDLC but they are combined and shortened.
  - Focused on functional and user interface reqs at the expense of detailed business analysis and concern for system performance issues.

Typical System Analyst Roles and Skills

- Analysts must have various skills which can be broken down into six major categories: *technical*, *business*, *analytical*, *interpersonal*, *management and ethical*.
- Analysts must have the **technical skills** to understand the organization's existing technical environment, the technology that will make up the new system.
- **Business skills** are required to understand how IT can be applied to business situations and ensure that IT delivers real business value.
- **Interpersonal**, because they often need to communicate effectively with users, business managers and programmers. They must be able to give presentations to groups and write reports. They also need to **manage** people whom they work and they need to manage the pressure and risks associated with *unclear* situations.

Typical System Analyst Roles and Skills cont.

- In big enterprises, it is not realistic to expect all these from a *single person* or a *role*. That is why they divide it to many roles.
  - Business analyst
  - System analyst
  - Infrastructure analyst
  - Change Management analyst
  - Project Manager

Business Analyst

- Focuses on the business issues surrounding the system such as:
  - Identifying the business value that the system will create
  - Developing ideas and suggestions for how the business processes can be improved
  - Designing new processes and policies
- Probably have business experience and training.
- Represents the interest of the project sponsor and the end users.
- Assists in the planning and design but *most* active in analysis phase.

System Analyst

- Focuses on the Information System issues.
  - Develops ideas and suggestions for how information technology can improve business processes.
  - Designs new business processes
  - Designs the new information system

Project Manager

- Responsible for ensuring that the project is completed on time and within budget.
- Manages the team members,
- Develops the project plan
- Assigns resources
- And is the primary point of contact when people outside the team have questions.
- Works in every part of the project.