Software Engineering

“At the end of this sequence, you will submit to your tutor a one-page summary of your research in the fields of software engineering to help you formalize what you have learned.”

“Concerning the expected report, provide one per group, go for up to 2-3 pages. Topics to be investigated:

- *Versioning tools, like git or SVN + their related hosting platforms, having devops in mind.*
- *Project management approaches and tools, Jira, Trello, etc.*
- *Communication tools like Teams, Discord, ...*
- *Development environments, like Visual Studio, the frameworks and debugging for instance.*
- *UML.*
- *Any other topic of interest, software engineering being a fairly broad concept.*

Content serves as an introduction and should cover a summarized explanation, a short statement about the relationship with your project, and possibly insights or feedback.“

What is Software Engineering ?

A software engineer is a person who designs, builds, tests, and maintains software systems. They use their knowledge of programming and engineering principles to create applications, websites, and other programs that people and businesses use every day. In other words, they plan how the code should be structured, how different parts of a system should interact, and how to make the software reliable and easy to maintain.

Software engineers use programming languages like Python, Java, C++, and many others to write instructions for computers. However, writing code is only part of the job. They also think about how to organize the program so that it runs efficiently, avoids bugs and crashes, and is easy for other engineers to understand and update later. They often create detailed designs (like UML diagrams) before even starting to code.

What tasks do Software Engineers do ?

There are two main types of software developers : Applications and Systems. Application developers work on products that are ready to use for consumers, internally or online. They go through the designing and conception of the software, including back-end, UI, or databases; but also maintaining functionalities, updating and fixing bugs. Systems developers, on the other hand, focus on operation systems (OS : Windows, Linux…). Most of the time their work is to develop a tailored solution for a specific customer. They are responsible for the conception, deployment and maintenance of those systems.

Successful software engineers use programming languages, platforms, and architectures to develop everything from computer games to network control systems. In

addition to building their own systems, software engineers also test, improve, and maintain software built by other engineers.

Software Engineering career :

Software developers can work in many places, as computer science is needed almost everywhere, from healthcare to the entertainment industry. There is the possibility to work for a software development company, a non IT company that needs to develop a digital solution for one of their sectors, or free lance for a variety of clients. Depending on this, the developer can either work alone or as part of a bigger development team.

Usually, to make a career in software development you need to have a formation in Computer Science. This can be done by going to an engineering school, public school, or if you're motivated by self-teaching.

What Skills do Software Engineers Need?

Since Software Engineers design and develop softwares to solve problems or simplify tasks, they need both technical and soft skills. Most of the jobs require either a bachelor in computer science, software engineering or related degree. For example, here are some skills and knowledge that might be needed:

- Coding Languages like Java, Python, C#, C++, …
- Object Oriented Programming
- Database management
- Operating Systems
- Versioning tools
- Collaboration
- Attention to detail
- Problem-solving
- Project Management

What are Versioning tools and what are the differences between them ?

A versioning tool can be defined as a software that helps in tracking and management of code. It will keep track of various versions of files and supply mechanisms to merge changes made by different team members.

The main objectives of version control tools are to maintain a history of changes, enable collaboration, facilitate the identification of who made specific changes, and enable developers to revert to previous versions if needed.

Here is a list of the most popular versioning tools:

- Git. GitHub is a web-based platform that uses Git. To simplify it, GitHub improves the usability of Git by supplying a centralized platform for hosting repositories, managing collaborative work, and providing additional features. There are also Git-based

alternatives, such as GitLab, Framagit, SourceForge.net, etc … However GitHub is the most popular choice for companies.

- SVN (Subversion). SVN or Subversion, is an open source version control system. Its principle is to host a project's source code on a central server and make it accessible to people who want to work on it. Each user has a local working copy of a version of the repository, often the latest one, in order to develop and make changes. Once modifications are made, the user submits them to the repository so that other developers can retrieve them.
- CVS (Concurrent Versions System). CVS, or Concurrent Versions System, is one of the oldest open-source version control systems. Its principle is similar to SVN: it hosts a project's source code on a central server, and users access it to work on the files. Each user retrieves a local working copy, makes modifications, and then commits the changes back to the central repository. CVS introduced basic concepts like concurrent work on the same files, but it has limitations in handling complex operations such as branching and merging, which led to the development of more modern systems like SVN and Git.

While Git, SVN, and CVS are all version control systems, they work differently. Both SVN and CVS are centralized systems: there is a single central repository, and users work by synchronizing their local copies with it. However, CVS is older and has more limitations, especially in handling complex operations like branching and merging, which SVN improved upon. In contrast, Git is a distributed version control system: each user has a complete copy of the entire repository, including its full history. This allows Git users to work offline more easily and enables more flexible workflows. Git also tends to be faster and more robust for most operations compared to SVN and CVS.

Project Management approaches :

There exist a lot of different approaches for project management. Some points are to consider before deciding on an approach. You need to consider these points:

- Size of the team
- What type of project
- Project difficulty
- The objectives of the project
- Team preferences (if the team prefers a visual board, …)

After all these points, you can consider one of the many different project management methods. Here are the most commonly used among companies:

- Agile: The Agile method is an iterative approach that focuses on flexibility and collaboration. Projects are broken into smaller cycles called "sprints," allowing teams to adapt to changes and continuously improve. It promotes constant feedback and customer involvement throughout the project.

- Waterfall: The Waterfall method is a linear, sequential approach where each phase of the project must be completed before moving to the next. It's well-suited for projects with clear, fixed requirements and minimal changes. Once a phase is finished, it's difficult to go back and make changes.
- Scrum: Scrum is a subset of Agile, focusing on delivering work in short, time-boxed iterations called "sprints," typically lasting 2-4 weeks. It emphasizes roles like Scrum Master and Product Owner, and daily meetings called stand-ups to keep teams aligned. Progress is tracked with visual boards and burndown charts.
- Kanban: Kanban is a visual project management method that uses boards to display tasks and their statuses in real-time. It aims to optimize flow by limiting the number of tasks in progress at any given time, ensuring efficiency. Kanban is flexible and can be applied to various project types, including those with continuous delivery.
- …

An overview on the different management tools :

Management tools are systems that help teams plan, organize, track, and collaborate on their work. They make it easier to manage tasks, deadlines, code changes, and communication.Examples include Jira (for tracking tasks and bugs), Trello (for project organization), and GitHub (for managing code and teamwork on software projects).

In Jira, you can create issues (which are like tasks or problems to solve), assign them to team members, set deadlines, and track progress. Teams usually organize their work using boards. For example, a Kanban board shows tasks as cards moving through stages like “To Do”, “In Progress”, and “Done”. Jira is especially popular for teams following Agile methods, like Scrum or Kanban, because it supports sprints, backlogs, and story points.

Nowadays, tools like github are more than just code repositories, they can have issues management, deployment, or more.

Communication tools :

There are quite a lot of different communication tools available. For a company, most of the time they will use Microsoft Teams, Zoom, Google chats, …

It really depends on the company and the need, but most of these can be used for any team or project.

Development environments :

“Development environments” refers to Integrated Development Environment (IDE).

There are numerous IDEs used, depending on the language or type of project. Some IDEs are made for several languages instead of only one, thanks to plugins. They usually share some features:

- Version control : Integration of interactions with Versioning tools, such as Git
- Debugging : A debugger is usually included, so that the user may add breakpoints to check the state of a code during execution
- Code search : Searching for specific code snippets inside of a project
- Syntax Highlighting : Making the code easier to read
- Code Completion : autocompletion of code being written, which allows to speed up repetitive tasks and reduce syntax errors
- Visual programming : Some IDE can create flowcharts or diagram, usually based on UML

Some common IDE choices :

- Visual Studio
- JetBrains
- Vim

UML :

UML, Unified Modelling Language, is a modeling language made to standardize the representation of the conception step of a project. It consists in a set of standardized diagrams that is used by developers to visualize and document the design of a product.

The most used diagrams are :

- Class diagram : represents interactions between classes and their structure for any object-oriented project
- Use case diagram : shows the interaction between the user and a feature, in their environment
- Sequence diagram : shows interactions of objects between in each other, and their order of occurrence
- Activity diagram : represents the workflow of the algorithm and all the actions of the programm