Software Development and Professional Practice

Introduction to Software Development

By

Dr. Rasha Mahmoud Kamel

What is Software Development?

- ***Software development*** is the process of ***taking a set of requirements from a user*** (a problem statement), ***analyzing them***, ***designing a solution to the problem***, and then ***implementing that solution on a computer***.

- ***Programming*** is really the ***implementation part***, or possibly the design and implementation part, ***of software development***. Programming is central to software development, but it is not the whole thing.

- ***Software engineering*** involves a process and ***includes software development***, but it also ***includes the entire management side*** of creating a computer program that people will use. It includes project management, configuration management, scheduling and estimation, managing people, and several other things.

What is Software Development?

- ***Software development*** is the fun part of software engineering.
- So, ***software development*** is a narrowing of the focus of software engineering to ***just that part concerned with the creation of the actual software***. And it is a broadening of the focus of programming to ***include analysis***, ***design and release issues***.

Ways to Learn Software Development

- Developing software is ***hard***. Learning how to develop software ***correctly***, ***efficiently***, and ***beautifully*** is also ***hard***.
- There are several ***ways to learn software development***:

(1) Reading excellent designs and **a lot of code:** By reading high-quality code and efficient designs, you:

- See how experienced developers structure solutions.
- Learn good naming practices, organization, and coding style.
- Observe efficient and elegant ways to solve problems.
- Learn how problems are broken down into smaller parts.
- Understand why certain design decisions are made.

Ways to Learn Software Development

(2) Writing a lot of code: This allows you to

- Experiment with the styles and examples you have seen in your reading.
- Try different approaches to solving problems.
- Make mistakes and learn how to fix them.

This process helps you develop fluency, discover what works and what does not, and internalize coding and design techniques.

(3) Thinking deeply about how you approach a problem and design a solution for it: This reflection allows you to examine how you work and how you design solutions, and to extract from your labors those patterns that work best for you.

How to Develop Software?

- To do software development well you need the following:

  (1) ***A small, well-integrated team.***

  - Small teams have ***fewer lines of communication*** than larger ones. It is easier to get to know your teammates on a small team. You know your ***teammates' strengths and weaknesses***, ***who knows what***, and ***who is the go-to guy*** for particular problems or tools.
  - Well-integrated teams have usually worked on several projects together. They are ***more productive***, better at ***holding to a schedule***, and more likely to ***produce code with fewer defects at release***.
  - The key to keeping a team together is to give them interesting work and give them the freedom to decide how to do it.

How to Develop Software?

(2) ***Good communication among team members.***

- Constant communication among team members ***is critical*** to ***day-to-day progress*** and ***successful project completion***.
- Teams that are ***co-located are better at communicating*** and communicate more than teams that are distributed geographically (even if they are just on different floors or wings of a building).

How to Develop Software?

(3) ***Good communication between the team and the customer.***

- Communication with the customer ***is essential to control requirements*** during a project.
- ***On-site*** or ***close-by customers*** allow for constant interaction with the development team.
- Customers can give ***immediate feedback on new releases*** and be involved in creating system and ***acceptance tests*** for the product.

How to Develop Software?

(4) ***A process that everyone buys into.*** Every project follows a process:

- ***Larger projects and teams*** tend to be more ***plan-driven,*** following processes with ***stricter rules and more documentation required***. They require more coordination to ensure that everyone knows what others are doing, preventing work conflicts or duplication. Such projects also require tighter controls on communication, using formal channels, and on configuration management, to ensure that all changes are tracked, approved, and integrated correctly.
- ***Smaller projects and teams*** tend to follow ***agile development*** processes with ***more flexibility and less documentation required***. This does not mean that there is no process; rather, it means doing what makes sense for the project to ***satisfy requirements***, ***meet the schedule***, and ***deliver a high-quality product***.

How to Develop Software?

(5) ***The ability to be flexible about that process.***

- No project ever proceeds as you think it will on the first day.
- ***Requirements change***, ***people come and go***, ***tools do not work out*** or get updated, and so on.
- This point is all about ***handling risk in your project***. If you identify risks, plan to ***mitigate them***, and then have a ***contingency plan*** to address the event where the risk actually occurs, you will be in much better shape.

How to Develop Software?

(6) ***A plan that everyone buys into.***

- You shouldn't launch a ***software development project without a plan***.
- The project plan ***encapsulates what you are going to do*** to implement your project. It talks about process, risks, resources, tools, requirements management, estimates, schedules, configuration management, and delivery.
- It doesn't have to be long and ***it doesn't need to contain all the minute details of the everyday life of the project***, but everyone on ***the team needs to have input into it***, they need to ***understand it***, and they need to ***agree with it***.
- Unless everyone buys into the plan, you are doomed.

How to Develop Software?

(7) ***To know where you are at all times.***

- It is that communication thing again. Most projects have ***regular status meetings*** so that the ***developers can "sync up" on their current status*** and get a feel for the status of the entire project. This works very well for smaller teams (say, up to about 20 developers).
- Many ***small teams will have daily "stand up" meetings*** to sync up at the beginning of each day. ***Agile processes*** often require daily meetings to ***improve communications among team members*** and to ***create a sense of team spirit within the team***.
- ***Plan-driven models*** do not require these meetings, depending on the ***team managers to communicate with each other***.

How to Develop Software?

(8) ***To be brave enough to say, “hey, we’re behind!”***

- Nearly all software projects have schedules that are ***too optimistic at the start***. Software developers tend to be optimistic, and this is reflected in their estimates of work: “Sure, I can get that done in a week!” “I’ll have it to you by the end of the day.” “Tomorrow? Not a problem.”
- ***At some point, you will be behind,*** and the best thing to do is to ***inform your manager right away.*** The important part is to keep your manager informed.
- ***The earlier you figure out you are behind schedule, the more options you have.*** These options may include ***extending the schedule*** (unlikely, but it does happen), ***moving some requirements to a future release***, ***getting additional help***, and so on.

How to Develop Software?

(9) ***The right tools and the right practices for this project.***

- One of the strengths of software development is that ***every project is different***. You need to examine each project and select ***the right set of development tools***.
- The three most important factors in choosing tools are the ***type of application being developed***, the ***target platform***, and the ***development platform***. Once these factors are clearly identified, developers can pick tools that improve productivity. The fourth important factor in tool selection is ***the experience of the development team***. If the team consists of experienced developers, tool choice is generally easier. However, if the team includes many fresh-outs and the target platform is new to all members, tool selection must be done carefully and additional time should be allocated for training and practice with the new tools.

How to Develop Software?

(10) ***To realize that you don't know everything at the beginning of the project.***

- Software development projects just don't work this way. You will always ***uncover new requirements***, other requirements will be discovered to be ***not nearly as important as the customer thought***, and others that were planned for ***the next release become top priorities*** This is known as churn.
- ***Managing requirements*** churn during a project ***is one of the single most important skills*** a software developer can have.
- If you are using ***new development tools***, you will ***uncover limitations*** you weren't aware of and ***side-effects that cause you to have to learn***, for example, ***three other tools to understand them***.

Thank you