Software Engineering Theory and Practice

School of Computing	
Title	Software Engineering Theory and Practice
Module Coordinator	Steven Ossont
Email	steven.ossont@port.ac.uk
Code	M30819
Moodle	https://moodle.port.ac.uk/course/view.php?id=11429

M30819: Software Engineering Theory and Practice

https://moodle.port.ac.uk/course/view.php?id=11429

Revision

- Software process models
- Requirements engineering
- Software design
- Software implementation
- Software testing
- Software maintenance

Exam Revision - Software Process Models

What is a *software process model*, and why do we need these models?

Software process models are defined as sets of activities teams follow to produce a software artefact. Models bring structure to the development process, and allow the development process to be planned and, most importantly, replicated.

Think of these models as *recipes* for developing a software artefact. They would tell you what activities are needed to develop the artefact, the order of these activities, their outputs, and how to move from one activity to the other.

Is there a *correct* model?

Several models exist and are usually followed. Some examples include waterfall, incremental, iterative, or agile. There is no such thing as the *correct* model. However, some models work better in some contexts than others. The choice of model relies on all sorts of factors, from the resources available (eg. time, end users, software engineers) to very specific standards imposed by external parties (eg. government bodies).

What is common, what is different?

All these models are based on the same set of activities:

- Specification - this is where we gather, specify, and validate the requirements.
- Design - this is where we build models of the artefact
- Implementation - this is where we translate the design models into source code
- Testing - this is where we test the code
- Maintenance - this is the part where we maintain and evolve the artefact over time
- Documentation - this is what we do all throughout the development process for all the other activities; we want to be able to remember decisions made and/or to let others know about decisions made
- Configuration - all the outputs of all of the activities will have multiple versions and will evolve over time; we need to keep track of all those versions and their evolution over time

Now, what makes the difference between these models?

The order in which we follow these activities.

Waterfall

Waterfall implies that we will follow all the activities above in a sequential order. So, once we have specified and validated a set of requirements, those will be set in stone, and we will move on to designing all the models up-front considering all the requirements. We will translate all these models into code, while testing units of code as they are written. We'll move on to integrating all units of code, test the system as a whole, and deploy it.

We'll speak to the end-users/clients at the very beginning to gather their requirements and at the end when showing them the end result.

We can't really go back one activity to change anything if we realise something's not right, the only time we can go back is at the very end of the process once we have an integrated system ready for maintenance.

Tasks

1. Think about the pros and cons of this model - when would it be the right model to use, and when wouldn't it?
2. How would you apply the waterfall model for the mock exam scenarios available ?

Incremental

Incremental approaches focus on specification first. Then, they look at breaking down the set of requirements into subsets. They will then focus all the other activities (design, implementation, testing, documentation, configuration) on each subset at a time. So, instead of designing models for all the requirements of the system, you'll really only focus on a subset of requirements and design a model to cover those. You will implement and test that model all while documenting and configuring the process. Once one subset of the requirements has been considered, you will have one increment of the system developed, so you can move on to another subset of requirements and follow the same process again until all the requirements have been covered.

For every increment, you will need to define the subset of requirements to be considered.

If, as a result of testing the system, new requirements are defined, you will need to add them to the specification. Once added to the pool, they will be considered for further increments just like all the other requirements would.

Tasks

1. Think about the pros and cons of this model - when would it be the right model to use, and when wouldn't it?
2. How would you apply an incremental model for the mock exam scenarios available ?

Iterative

Iterative approaches look at the sequence (specification, design, implementation, testing) (or variations of this sequence) as a cycle which is first applied to develop a basic version of the complete artefact. Then, gradually, the same cycle is used to refine that initial version and all the versions following after that. So, you would specify, design, implement, and test an initial version of the system. Then, you would go through the same sequence of activities to refine that version until you get to the software artefact your client is happy with.

For each iteration, you will need to define the scope of the version to be developed.

Tasks

1. **Think about the pros and cons of this model - when would it be the right model to use, and when wouldn't it?**
2. **How would you apply an iterative model for the mock exam scenarios available ?**
3. **Think of possible variations of these approaches, such as an incremental-iterative approach, and how this would be applied for the mock exam scenarios available .**

Tips for answering the exam questions on Software Process Models:

- Start by sketching the generic model specified by the question on a separate sheet of paper. Be very clear on what development activities are involved, and what is the order/sequence of these activities.

Tips for answering the exam questions on Software Process Models:

- Think carefully about how each of the activities identified will be put in place for developing the system defined in the scenario provided to you. Perhaps you'll need to consider other sub-tasks, or maybe you will need to focus on specific outputs.

Tips for answering the exam questions on Software Process Models:

- It is really important to make sure that these activities and their sub-tasks are specific to the system defined in the scenario provided to you. Generic tasks such as *design models* or *specify requirements* will not be considered as valid answers.

Tips for answering the exam questions on Software Process Models:

- For incremental approaches, identify the requirements to be considered as part of each increment.
- For iterative approaches, identify the scope of the version to be developed as part of each iteration.

Tips for answering the exam questions on Software Process Models:

- When representing the order in which these activities will be followed, you can use any format you wish. Gantt charts are an option, but you are free to use any other formats - from a simple table to anything more elaborate.