Here you can write a question to SOE. I will check in a few times before 2019, and see if I can answer any questions raised.
Carsten R. Jakobsen

Ian Sommerville writes that an incremental process can be plan-driven (p. 50). Can you provide an example of one?
Answer: The key to understand a plandriven incremental approach, is to understand that the increments are defined *ahead of time*. Often most or all requirements are established up front, and then the development of the system is sliced into increments. A first increment in a system could include functionality to add and remove users. A second increment could offer a core functionality, e.g. the ability to login and see the balance on my bank account. A third increment would allow me to transfer money to another account. The key is, that the content of these increments are specified ahead of time. The learning from the increments are used to confirm that the overall plan with all increments will be met on time, cost and schedule. All increments are allocated budget ahead of time. As the project completes each increment, it can be observed if increments are produced to expected cost. Assume the project is sliced into 8 increments, and the first 3 increments are now complete. If the first three increments are using 25% more cost than estimated, it is likely that the remaining 6 increments will also use 25%, unless some action or improvement are taken.

Another example of incremental, is RUP, that can be considered a combination of incremental and iterative. RUP has four phases, that each can be considered an increment: Inception, Elaboration, Construction and Transition. Each phase usually contains work from the disciplines: requirements, analysis, design, implementation and test. Please read the text from Larman on the purpose of each phase (Text is part of lecture 3).

We are struggling to find the material related to exam topic 8 (Managing change to requirements), where can we find this?
This is described in Sommerville chapter 2.3 (Lecture 1) and 23.2 (Lecture 5). You can find slides in lecture 1 slide 30-33, lecture 5 slide 8. Also at lecture 5 you will find two pictures related to agile and plan driven planning. 4.3 (Lecture 12) on how to elicit requirements may also give some inspiration.Finally Sommerville chapter 8.4 suggest how user test or acceptance test csm contribute to changes that must be managed.

When is the deadline for submitting part 3 of the miniproject?
You are only supposed to submit the miniproject at lecture 4 and lecture 9 (part 1 and 2).

Will a PC and/or whiteboard be available at the exam to use for our presentations?
It’s an oral exam. There will be a white board. PC will not be available

What if you did not submit the miniproject in time?

Check the exam list. To my knowledge all of you are admitted to the exam.

In the readings list, Boehm: A spiral model of software development and enhancement is listed, but I can't find it on the moodle page? Furthermore, as far as I could find, the spiral model is only to be found on the last slide of lecture 6.
This is an error. As a general rule, you will only be examined in topics you have had the opportunity to learn. During lecture 6 this includes what is presented on the slides and on lecture 12 the content of the slide: Other topics: Spiral model. **NOTE: The article is now available on lecture 6 in Moodle.**

Do you think it is a good idea to use the whiteboard at the exam? e.g. starting of topic one by drawing the waterfall model on the whiteboard and explain it from that.
It is very different what works for different people. You are welcome to stand up and use the white board, or sit down and talk or both - as you please. I recommend that you structure your initial response around the agenda proposed on slide 5 at lecture 12:

1. What are the central concepts in this topic
2. What are advantages and disadvantages
    a. Any alternatives, if yes, how to choose
3. What are related topics

In the slides from lecture 12 you say that the spiral model is a tool to organize requirements. Can you explain that? To my understanding the spiral model is related to risk management.
The spiral model has strong connections to risk management.
The key driver in the spiral model is risk management. This is used for short iterations on concept of operations, requirements and design, to allow for an early and efficient clarification of what are the right requirements and design. The very last part of the spiral model includes the traditional phases from waterfall: detailed design, code, unit test, integration test, acceptance test, Implementation.

The spiral model was a response to a number of other models used at the time:

- Code and fix: write some code, then fix some errors, code again. This model ends up in spaghetti code, and showed the need for a: a) design phase b) requirements phase c) early planning and preparation of test
- Waterfall model. The disadvantage is that it is very document driven, and some classes of software are difficult to drive with documents, e.g. interactive end user applications (here prototypes are a much better way to validate requirements).
- The evolutionary model: expanding increments of a software product. Could resemble to code and fix model, with the risk of ending up with hard-to-change code.

The spiral model proposes to address the disadvantages of these model, by using risk analysis in the early phases. It reflects that software changes become exponentially more expensive, the

later changes are identified. It costs much less to change a requirement, than a design, and even more to change when the code is written. Based on risk analysis an appropriate mix can be made of specification oriented (waterfall like), prototype-oriented (evolutionary-like), simulation-oriented approach to software development. Therefore Boehm claims that the spiral model can be adopted to any of the previous models.

The spiral model should be read in this way. The radial dimension represents cumultative costs. The model starts in (0,0) where no cost is yet accumulated. It works in cycles divided into steps. Each cycle starts with risk analysis and end with a review - and includes appropriate steps of prototype, (simulation, models or benchmarks). The three first cycles cover 1) concept of operation 2) validated requirements 3) Validated design and verification.

The spiral model iterates in the beginning to ensure that we get the right requirements. This is different from agile methods, where the assumption is that no matter what, we will guess the wrong requirements initially, so instead agile works to maximize learning by delivering working software in small chunks as fast as possible.

Please let me know if this clarifies, or if you have additional questions to this answer.

Thank you, this clarified it well.

In lecture 7, Plato and Aristotle came up in regards to their definitions of quality. I cannot find clear definitions of their views of quality in Dahlborn's and Mathiasen's "Artifacts". Can you? Either way, what is your take on them?
I wouldn't worry much about details of Plato and Aristotle. Instead I would make sure I understand that there are different types of quality as defined from 1) expectation 2) product or 3) process. The article describes expectation quality as a subjective quality. The debate between Plato and Aristotle can be considered as a discussion about: is there a divine (objective) quality in all things, or is quality a subjective experience.
Examples:

- the coffee pot from SAS (apparently dull, but lot of quality build into *product*)
- Mercedes car: I have many *expectations* of high quality to it (at least for me as a man)
- Bernaise sauce: I use the same *process* every time, same amount of ingredients, heating and whipping until a certain resistance or look. This same process delivers the same quality of sauce every time
-

During the preparation for the questions, I had trouble distinguishing "Q9: Quality Control" from "Q12: Quality Management". Can you clarify what separates these topics?
Sommerville ch. 24 Quality Management (This chapter is not curriculum, but the content of this paragraph has been part of the lectures)::
"Software Quality management techniques have their roots in methods and techniques that were developed in manufacturing industries, where the terms *quality assurance* and *quality*

control are widely used. Quality assurance is the definition of processes and standards that should lead to high quality products and the introduction of quality processes in the manufacturing process. Quality control is the application of these quality processes to weed out products that are not of the required level of quality. Both quality assurance and quality control are part of quality management.
In the software industry, some companies see quality assurance as the definition of procedures, processes, and standards to ensure that software quality is achieved. In other companies, quality assurance also includes all configuration management, verification, and validation activities that are applied after a product has been handed over by a development team".

Within Quality Management you have *quality assurance* and *quality control.* In quality control I will dive more into the details of test in quality management I will have more focus on planning of quality and where it is in the process. In practices there will be a huge overlap, since we will normally end up discussing both.

I have talked to it at lecture 7 slide 13. Assignment 16 a+b are about updating your plan/processes to plan quality management. In lecture 5 about project planning, this is reflected in Sommerville chapter 23 in table 23.2 project plan supplements, where plans for quality and validation are called out. At lecture 11 this is also addressed in assignment 19.

I think it boils down a very simple point: quality includes both planning of process/activities, and execution of process/activities. Waterfall uses documents for such plans, in agile it is more informal see lecture 7 last 2 slides.

Inkrementiel og iterativ står som om det er samme model, men vi mindes at der er forskel på dem. Et af spørgsmålene lyder også på om den inkrementielle model kan være iterativ.

Inkrementiel betyder at skære ting i skiver, hvor vi kalder hver skive for et inkrement. Forskellen på et inkrement og en prototype er at vi forventer at bygge videre på et inkrement, hvorimod vi ofte forventer at smide en prototype ud.

Iterativt betyder, at genbesøge / justere / iterere. Ud fra den læring/feedback vi har opnået i en iteration/inkrement, så ændrer vi indholdet af vores fremtidige arbejde (dvs efter inkrementet laver vi inspect and adapt). Vi kan have lært at det vi lige har implementeret, i virkeligheden skal laves på en anden måde - eller gennem det arbejde vi lige har lavet, kan vi se at noget af det arbejde vi oprindeligt troede vi skulle lave bagefter ikke længere er nødvendigt, og opdaget nye opgaver der skal tilføjes.

Scrum er inkrementiel og iterativ. Hver iteration er et inkrement, der kan leveres til kunde/bruger hvor PO ønsker det. Scrum laver dagligt (Daily Scrum) og ved afslutning af hvert inkrement (Sprint Review) inspect and adapt, og er dermed en iterativ proces. Scrum er en meget iterativ proces.

Vandfald er faseopdelt, og kernen i modellen er at hver fase laves helt færdig. Så det er big bang != inkrementiel.

Spiral modellen anvender iterationer i starten af projektet, for at sikre at det er de rigtige krav der arbejdes på. Der kommer ikke inkrementer af kørende software ud til kunden, men der er iterativ læring i krav og analyse fase. Spiral modellen slutter som traditionel vandfald med udtømmende design, implementering og test.

RUP har 4 faser der falder i sekvens. Inden for især de sidste tre faser er der inkrementer. I Første fase låses 10% af krav, i anden fase 80 %, så ud over at være inkrementiel, så tillader RUP i sin definition også muligheden for at inspect og adapt i begrænset omfang, og er dermed også en iterativ proces.

The Q&A session has ended.