Lecture "Software Engineering SoSe 2023

Free University of Berlin, Institute of Computer Science, Software Engineering Group Lutz Prechelt, Linus Ververs, Oskar Besler, Tom-Hendrik Lübke, Nina Matthias

Exercise Sheet 12 **Process Models & Project Management** to 2023-07-10

Task 12-1: Selection of process models

a) Briefly summarize again the differences between the three process model types *waterfall*, *evolutionary*, and *incremental* together.

b) For each of the following systems to be built, select the most appropriate process model type (from those listed in **a))** and justify your choice.

1. A terminal as a digital, interactive replacement for paper timetables in larger train stations (for arrival and departure times).
2. A control unit for the anti-lock braking system (ABS) in passenger cars.
3. A system for managing courses, such as our KVV.

Task 12-2: Waterfall vs. Agile Processes

a) Over the past fifteen years, the waterfall model has been heavily criticized and more "agility" was called for. Before that, however, a waterfall-like approach had (almost) always been described as the ideal approach. Explanation of this development:

1. Why, on the one hand, can the waterfall model be considered quite *ideal*?
2. And why, on the other hand, did people still break away from it? Name at least two points where waterfall projects can fail and which are at least strongly cushioned in agile projects.

b) Fill in the following cloze with appropriate terms. If necessary, use the sources provided on the lecture website.

In terms of time, a project carried out with Extreme Programming (abbr.) is divided into ______________divided. Each of these ends with a new version of the software system. At ____________________of each iteration the customer and the developers discuss together which functionalities should be realized. The customer formulates his wishes on the ____(engl.). During the entire development the customer is _______ __________________. He also defines ____to be able to test the ____________________ __________________to be able to test the functionality at the end of each iteration. For the developers, Extreme Programming additionally specifies a number of practices, such as
_____________, ________________or shared responsibility.

Task 12-3: Project planning

You run a small software company and have recently secured an order for a product development. This project involves the development of a cloud-based smartphone app that you want to develop completely without external service providers.

Your company is still relatively young, but you have a number of team-oriented employees who are able to work productively with each other and in every area of responsibility.

You have already completed the task decomposition and estimation (in weeks), which is important for planning:

ID	**Description**	**Duration**	**Predecessor**	**required persons**
A	Requirement survey server	3		2
B	Requirements elicitation client	2		2
C	Implementation client	6	B	1
D	Implementation server	3	A	1
E	Recruit test subjects	3		1
F	Procure server hardware	2		2
G	User tests and improvements	4	C, E	2
H	Load test on real hardware	4	D, F	1
I	Automate deployment	2	H	1
J	Going Live	3	G, I	1

Tasks should not be further divided here (neither in terms of content nor time); each developer works on at most one task. Tasks with cannot be started and worked on with less than the "required people"; additional people do not speed up the work.

Now it's time and resource planning.

a) Create a *network diagram that* shows the logical dependencies between the tasks. Represent each of the above tasks as a rectangle and enter the *ID*, the *task duration* and the *earliest possible* start.

b) Create a *Gantt chart that* makes the time dependencies of the tasks visible.

c) Determine the *critical path*(s), the *shortest project duration*, and the *slack time for* all activities.

Now that you have an overview of the logical dependencies, turn to the "resources", in this case the personnel.

In contrast to network plans and Gantt charts, this is a concrete assignment of activities to developers. In order not to lose the overview, you should choose a suitable representation, for example one like in Fig. 1.

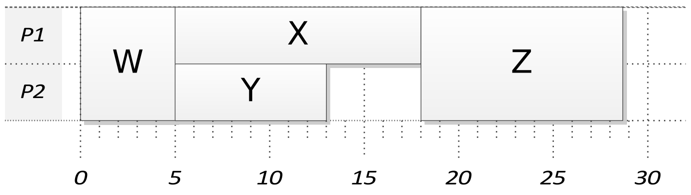

Figure 1: Graphical representation of a task distribution for two people: First, P1 and P2 work together on task W, before they then split up for tasks X and Y. Task Y takes a shorter time overall than task X, but P2 cannot yet start on task Z because it again requires two people.

Work on the following tasks:

d) Assume hypothetically that you want to parallelize the project process as best as possible (given the dependencies), i.e. you want to minimize the project duration. What would be the personnel requirement P_{maz} , i.e. the largest sensible team size above which additional persons no longer result in an acceleration?

Graphically represent a possible distribution of tasks for $n = P_{maz}$ (approximately as in Fig. 1). For this distribution, also indicate the buffer times of all tasks, as well as the project duration.

e) Suppose you could only assign two developers to this project. What would be the shortest project duration here?

Graphically represent a possible task distribution for $n = 2$. Specify the buffer times and the total runtime for this distribution as well.

f) Graph task distributions for all other possible team sizes n (i.e., $2 < n < P_{maz}$) so that in each case the project duration is as short as possible. Again, specify the project duration and the buffer times for each distribution.

g) Consider your possible project flows (i.e. for all team sizes from 2 to eventually P_{maz}) and make a decision: How many developers do you now put on this project?

Explain your decision: Compare your options explicitly according to at least three aspects.