Aufgabe 12

Nathan Ritter
5566519
Lorenzo Tecchia
5581906
2023.07.10

Contents

Chapter 1

Task 1

(a) In the waterfall process model the phases of building the software system are often separated and the cascading use for the next phase to rely on the previous one give the name to the process model. In contrast the process model "incremental" gives an interleaved approach between phases to the software building; the system is developed as a series of version and each version release goes through normal phases of software building and each version implements new functionalities to the previous one. Finally the evolutionary model, fuses the model of "iterative" and "incremental" to first construct what could be described as the skeleton of the software system; in this model the processes of creating the software and maintain the software "collide" and the software evolves in response to new requirements with the passage of time, is especially used in large projects. [3][4]

(b) These are the choices:

1. The evolutionary model, since a "skeleton" of the concept of the system already exists, the paper one, the latter has to be used as a stepping stone to evolve into the digital and interactive version of it.
2. Waterfall process model, since in this case the requirements don't really much evolve during the building of the system, the ABS like its control unit should serve just one and only one purpose, and also the whole system should be built in its entire functionality before starting any other process forward (e.g. V&V). Falls into the more traditional mechanical process of building something.
3. Incremental model, since the version of for example the KVV can be released and students that use it can give back valuable feedback about it and help the system to adapt new requirements and possible defects present into the system.[3]

Chapter 2

Task 2

(a) Comparison:

1. Since in the waterfall model the requirements set for the system are "frozen" the cost to develop the system itself stay constant and doesn't add app during development. Also By completing a full design early in the project, changes to systems stay minimal, meaning the cost to fix and alter designs is kept low. Adding to this by having detailed documentation and designs, a project can lose key members without too much hassle since the documentation describes in reasonable detail how any subject-matter expert of the product or skill are needed to complete the work. So it shows that the waterfall models can be used by roughly three types of systems:
   - Embedded systems: because of hardware restrictions no change into the software requirements can be made after implementation
   - Critical systems: since the safety is key word here, the process of security analysis must be thorough and the design documents must be complete to provide an effective analysis.
   - Large software system: where multiple partners and/or companies are involved. The system requirements must be unchanged to favour independent development, and the completion of such type of software and the withdrawal of any partner from the software should still guarantee the usability of the software afterwards.[1][3]
2. In the case of the waterfall model and in the context of a rapid changing world of requirements, by the time the software is available for use, the original reason for its procurement may have changed so radically that the software is effectively useless, instead the agile method would permit the process to adapt and to incorporate in an interleaved way the new requirements for the software.
   Also the waterfall model is not the right process model in situations where informal team communication is possible but instead the agile method permits as key principle: collective ownership in which developers work

> on all areas of the system, so that no islands of expertise develop and all the developers take responsibility for all of the code. Anyone can change anything.[3]

(b) Zeitlich ist ein Projekt, das mit Extreme Programming [5] durchgeführt wird, in Versionen eingeteilt. Diese enden jeweils mit einer neuen Version des Softwaresystems. Am Anfang einer jeden Iteration besprechen der Kunde und die Entwickler gemeinsam, welche Funktionalitäten realisiert werden sollen. Der Kunde formuliert dabei seine Wünsche auf den SUD (engl.). Während der gesamten Entwicklung ist der Kunde beteiligt. Er definiert zudem Akzeptanztests, um Ende am jeder Iteration die Funktionalität testen zu können. Für die Entwickler gibt Extreme Programming zusätzlich eine Reihe von Praktiken vor, wie etwa das nuzten von Code um Gedanken zu kommunizieren, das Respektieren von sich und anderen oder die gemeinsame Verantwortung.

Chapter 3

Task 3

Table 3.1: Reference Table 1

ID	**Description**	**Duration**	**Predecessor**	**Required People**
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 deploymen	2	H	1
J	Going live	3	G,I	1

Table 3.2: Reference Table 2

Earliest Start	Duration	Earliest Finish
Activity Label		
Latest Start	Slack/Float	Latest Finish

(a) Solution:

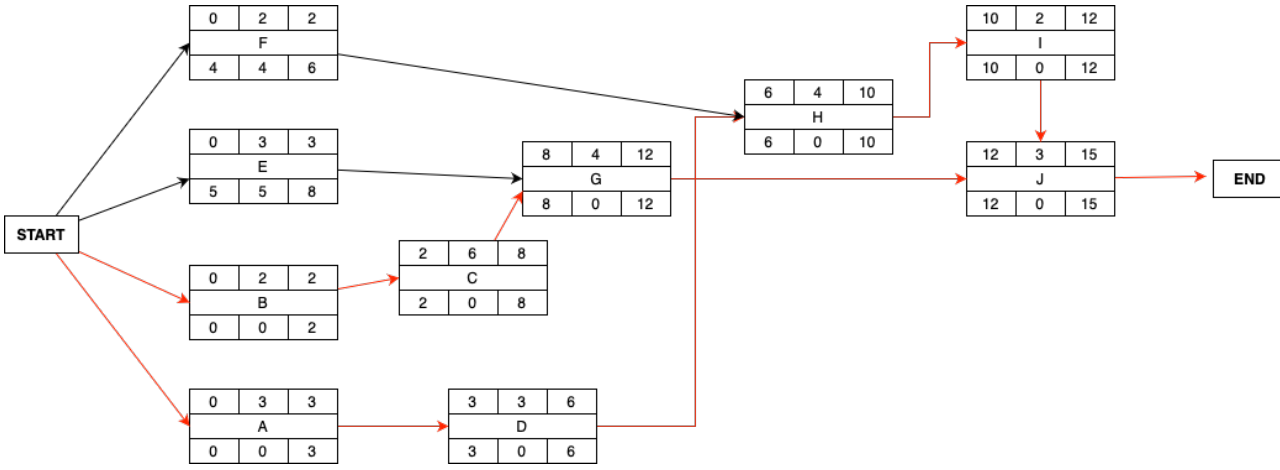

Figure 3.1: Task 1.a and 1.c

x

Earliest start = Latest start form previous activity (going from left to right)
Last finish = Earliest finish from previous activity (going from right to left)
Latest start = Latest finish - Duration
Slack = ES -LS or EF - LF
Critical path is formed when activities have 0 slack. (in red)
Shortest project duration is shown on the critical path: 15 weeks.[2]

(b) Solution:

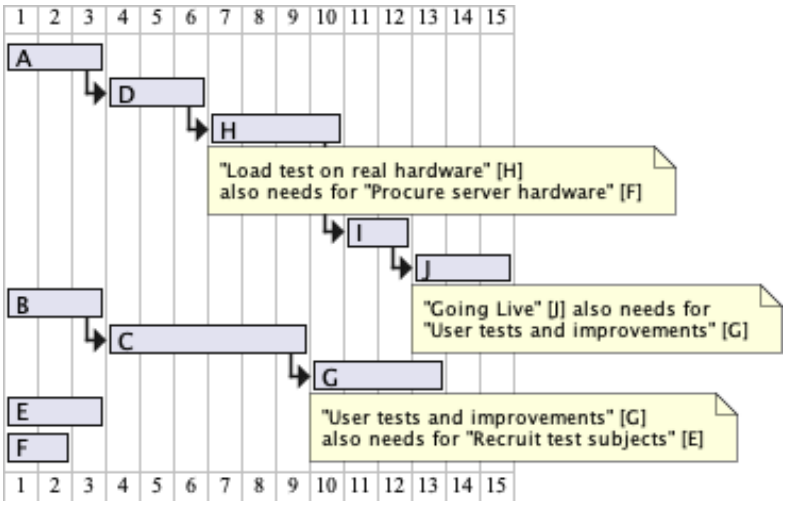

Figure 3.2: GanntChart

Bibliography

[1] Peter Reeves Aiden Gallagher Jack Dunleavy. *The Waterfall Model: Advantages, disadvantages, and when you should use it.* URL: https://developer.ibm.com/articles/waterfall-model-advantages-disadvantages/.

[2] *Software Engineering | Critical Path Method - GeeksforGeeks — geeksforgeeks.org.* 2022. URL: https://www.geeksforgeeks.org/software-engineering-critical-path-method/.

[3] Ian Sommerville. *Software engineering.* 10th ed. Boston: Pearson/Addison-Wesley, 2015. ISBN: 978-1-292-09613-1.

[4] Shreeya Thakur. *Evolutionary Model In Software Engineering (With Examples) // Unstop (formerly Dare2Compete) — unstop.com.* 2022. URL: https://unstop.com/blog/evolutionary-model-in-software-engineering.

[5] Wikipedia contributors. *Extreme programming — Wikipedia, The Free Encyclopedia.* [Online; accessed 10-July-2023]. 2023. URL: https://en.wikipedia.org/w/index.php?title=Extreme_programming&oldid=1150668036.