Individual Activity: understanding the software process through a system dynamics model

Course: 2019-2020

Facilitators: Dr. Maribel Sánchez-Segura, Dr. German Dugarte-Peña

Introduction.

As part of the ***Software Development Projects Management*** course, it is important to comprehend the whole process of software development and how the behavior of the different phases composing the development process may be, according to variations in factors affecting the performance of the project under development.

Following, you will find an explanation of a general software process behavior, the different phases of such a process, the relations among such phases and a slight mention of factors that may affect the process.

Next, you will be asked to reply to questions about possible effects of factors on the behavior of the phases of the process.

Please watch a video with the introduction to this exercise. Click here to watch the video

Objective

The main goal of this activity is to help you, to understand from a management perspective the dynamic behavior of the general software production process and how it may be affected according to variations of factors affecting the process.

Procedure

This individual assessment consists of four sequential sub-activities:

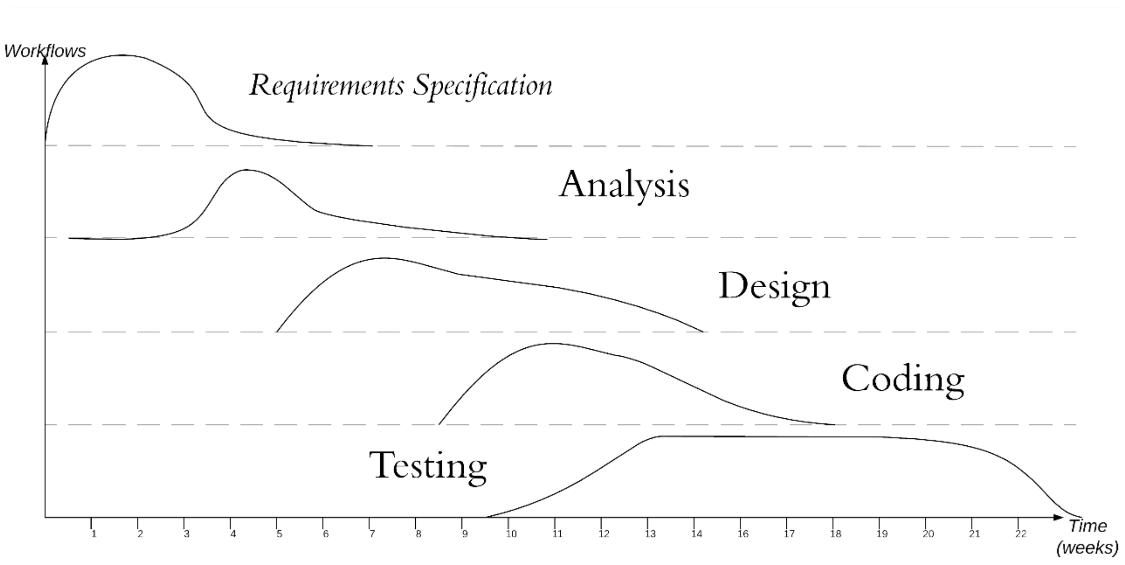
1) Explanation of the Software Process Dynamics. [To be read by your own before next session]
2) A first assessment of comprehending questions. [To be done during next online session through a web form]
3) Video: "System Dynamics and the Software Process".[To be done during next online session]
4) A second assessment of comprehending questions. [To be done during next online session through a web form]

More details on 1), 2), 3) and 4), are given below.

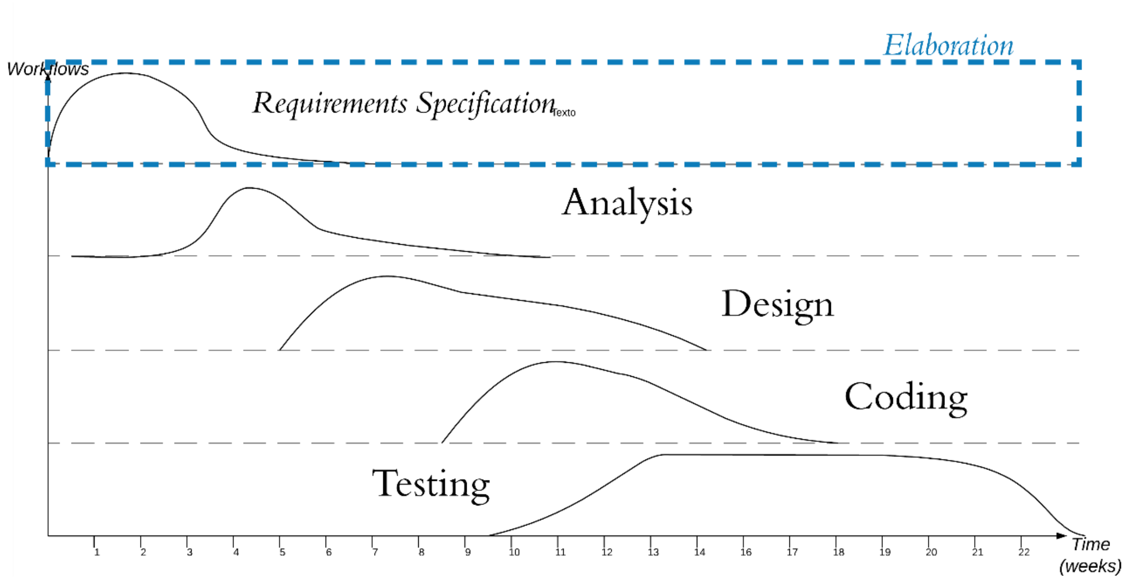
1) Explanation of the Software Process Dynamics: The process behavior

Software development process comprises several phases which are not only co-related but co-dependent. From the beginning to the end, a software development project comprehends a series of tasks that must be accomplished to be able to conclude the project. Each of these tasks helps to achieve one or several of the phases of a project, making as a whole the project to evolve.

According to Craig Larman's software development model (Larman, 2015), a project comprises three main phases: "**Planning and Requirements Specification**" (also called Elaboration), "**Construction**" and "**Installation**". "Planning and Requirements Specification" (or Elaboration) and "Installation" phases are performed only once, while the "**Construction**" phase is a made up of **several development cycles**, each of which contains: **Design, Analysis, Implementation, Plan Refinement and Model Synchronization**.

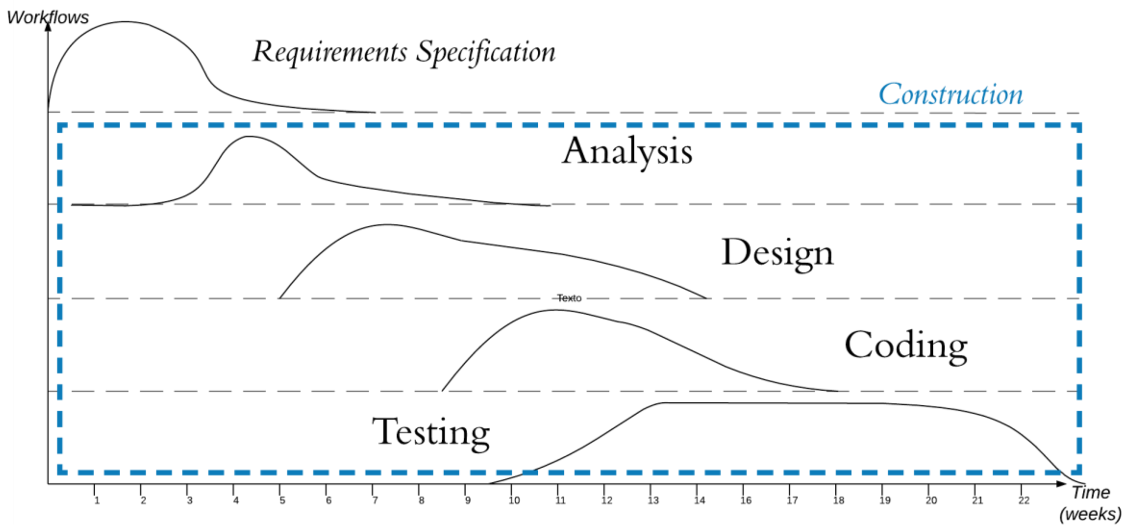


According to the description of Craig Larman, in any software process, the initial phase is the Elaboration, which consist mainly of the requirements specification, comprising the activities aimed at identifying what are the actual needs of the customers of the software product to be developed. It starts executing from the week/day/month 0 of the project, and takes some time, according to factors such as: the complexity of the product to be developed, the available information for requirements elicitation, the availability of stakeholders to elicit the knowledge about the contextual problems, and many other factors. In the following figure, the workflow on requirements specification concentrates on the first weeks of the project.



During this initial phase, there is very interactive activity between those actors that best know the problem or the situation in which the software is supposed to be used, and those actors of the development company who need to understand the software product related needs and express them in terms of requirements for the software to be developed.

Once that the requirements have been identified, it comes the second phase of the process: The construction, comprehending in this specific version of the Larman process: the Analysis, the Design, The Coding (also known as Implementation) and the Testing, as shown in the following figure.

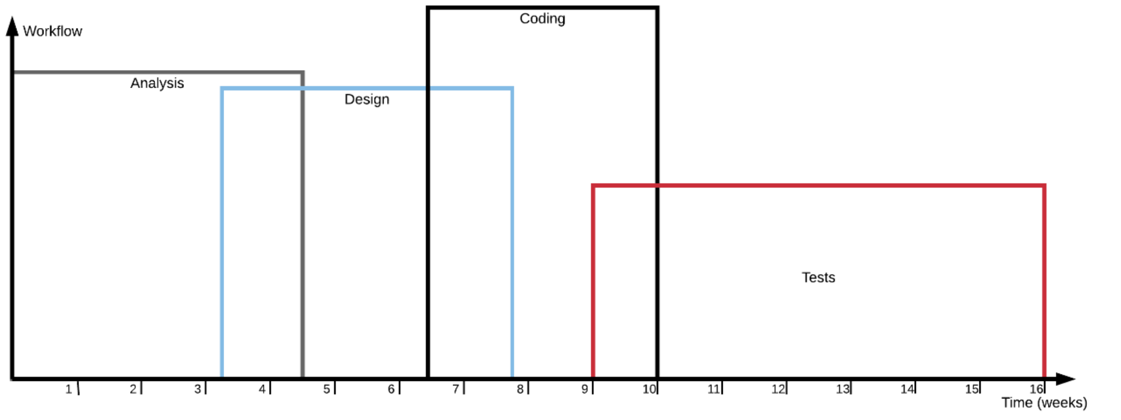


The construction phase comprehends four sub-phases: The *Analysis*, The *Design*, The *Coding* and *Testing*.

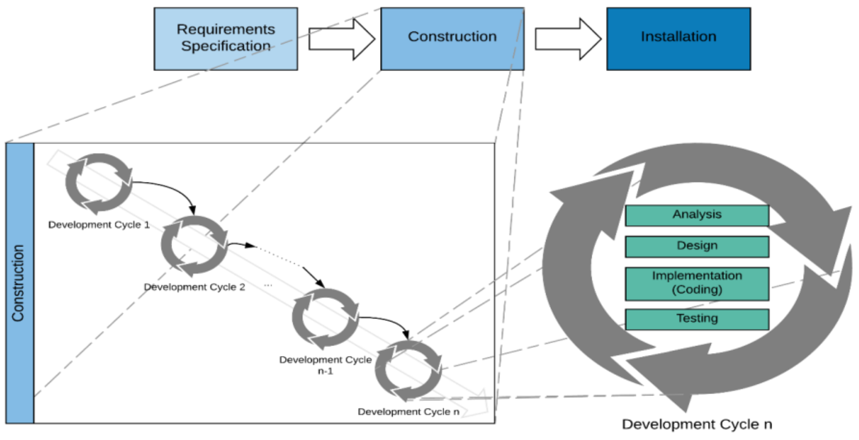
"Construction is the largest phase of the project. In this phase, the remainder of the system is built on the foundation laid in Elaboration. System features are implemented in a series of short, time-boxed iterations. Each iteration results in an executable release of the software. It is customary to write full-text use cases during the construction phase and each one becomes the start of a new iteration. Common Unified Modeling Language (UML) diagrams used during

this phase include activity diagrams, sequence diagrams, collaboration diagrams, State Transition diagrams and interaction overview diagrams. Iterative implementation for the lower risks and easier elements are done. The final Construction phase deliverable is software ready to be deployed in the Transition phase" (Wikipedia, 2019)

As you may be thinking, in the "**Construction**" phase, maybe there is a sequential process, which means that it is expected that **the "Analysis" must be finished before working on the "Design"**, and similarly, **the "Design" before the "Implementation"**, and the **"Implementation"** before the **"Testing"**. However, the truth is that **these phases overlap in time**. The accomplishment of the phases in one lifecycle may be represented as:



What you can see in the X axis is the time that it takes for an average project to complete its workflow for an iteration. According to the Larman Process, there is a pseudo-sequence as: Analysis-Design-Coding-Tests. In the Y axis you can see what is a representation of the workload for each of the phases of the software development process in each iteration. The gray line represents the **Analysis**, i.e., the first phase, which core activities start being done in reference to the whole process; the blue line represents the **Design**, the black line represents the **Coding**, i.e., the core implementation activities; and the red one represents the **Testing** phase. The following figure illustrates how each iteration comprehends the four sub-phases previously mentioned, constituting a development life-cycle. The whole process of software development comprehends several life-cycles that constitute the construction.



How may the Process be affected?

The whole software production process is affected by several elements during all the different phases and sub-phases, i.e., according to the COCOMO II estimation model (Boehm, 2012), there are Software, Hardware, Personnel and Project attributes that affect the way in which the software development phases are accomplished. Specifically, according to the COCOMO II project estimation model, the factors (also called drivers) affecting the process may be listed as:

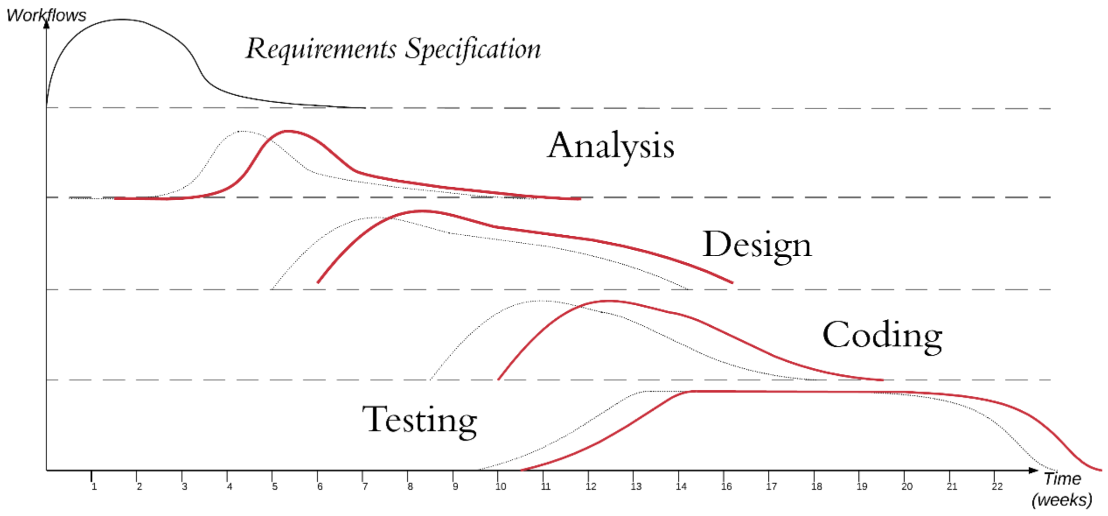
Type of attribute:	Factors or Drivers	Acronym
Software Attributes	● Reliability ● Database Size ● Complexity	● RELY ● DATA ● CPLX
Hardware Attributes	● Runtime restrictions ● Virtual memory restrictions ● Volatility of the virtual machine ● Response time	● TIME ● STOR ● VIRT ● TURN
Personnel Attributes	● Analysis capacity ● Application experience ● Quality of the programmers ● Experience in the virtual machine ● Experience in the language	● ACAP ● AEXP ● PCAP ● VEXP ● LEXP
Project Attributes	● Up-to-date programming techniques ● Use of software tools ● Development time constraints	● MODP ● TOOL ● SCED

Those factors, by default take a value of "1", which is called the nominal value, and it is the neutral one, meaning that a factor with value 1 is not affecting the project, neither for god or for bad. Each of these factors can move in a range of values, not necessary to be known for this individual assessment. When needed, the explanation of the factors will be provided.

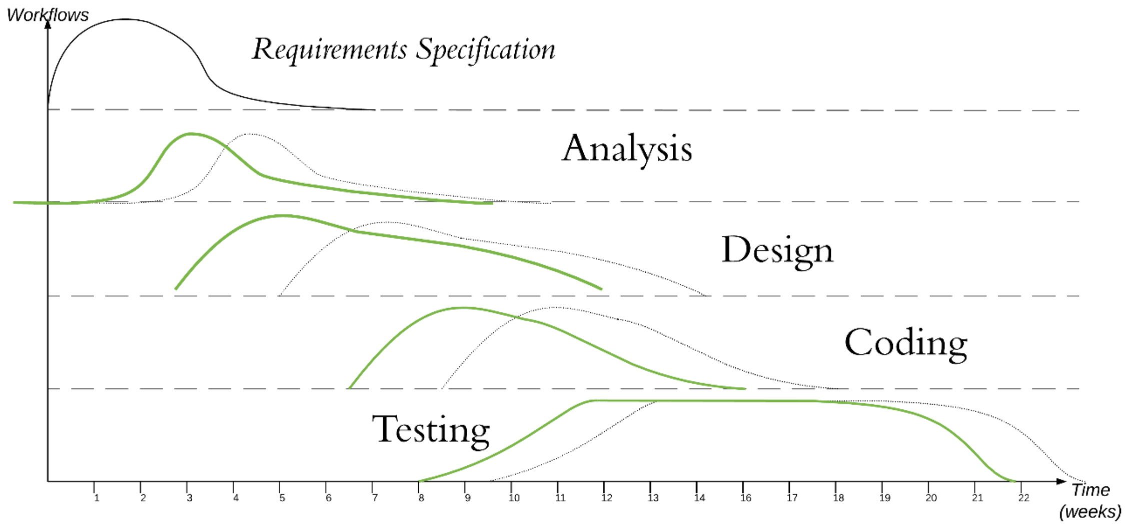
You may think about even more factors affecting the whole process, however, for this exercise, the list of factors was constrained to the COCOMO II's since it is a widely accepted model within the field of Software Economics and the software industry in general.

Effect of changes in attributes over the process

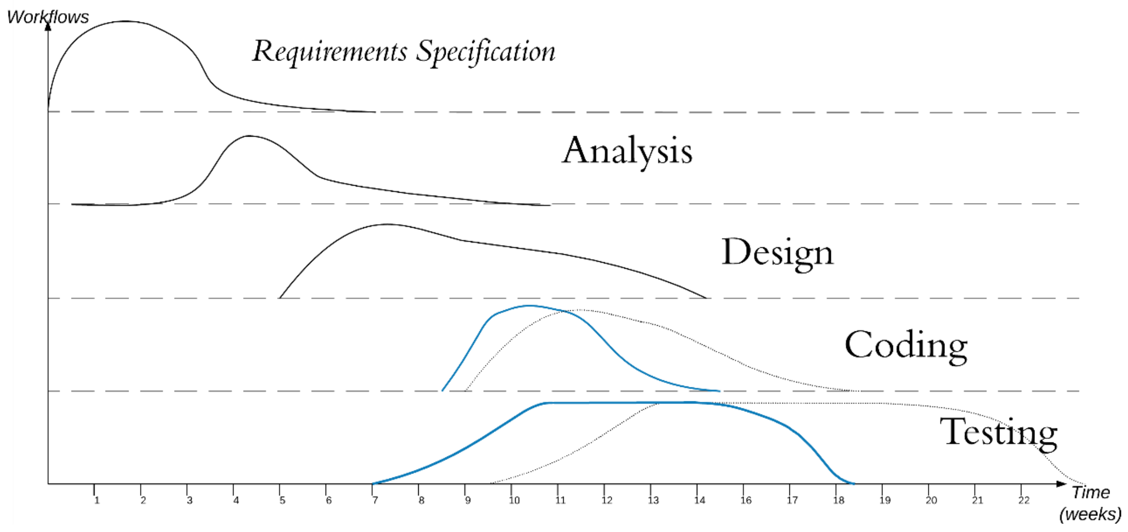
Let us suppose now that the capacity of the analyst (known as ACAP) is worse than in the nominal state. Such different valuation for the capacity of the analyst is supposed to have a knock-on effect on the whole process, i.e. the analysis sub-phase will take longer than in the nominal case. Similarly, and as a consequence of this, the Design, Coding and Testing sub-phases are expected to start later and take longer. (See the following figure illustrating these effects).



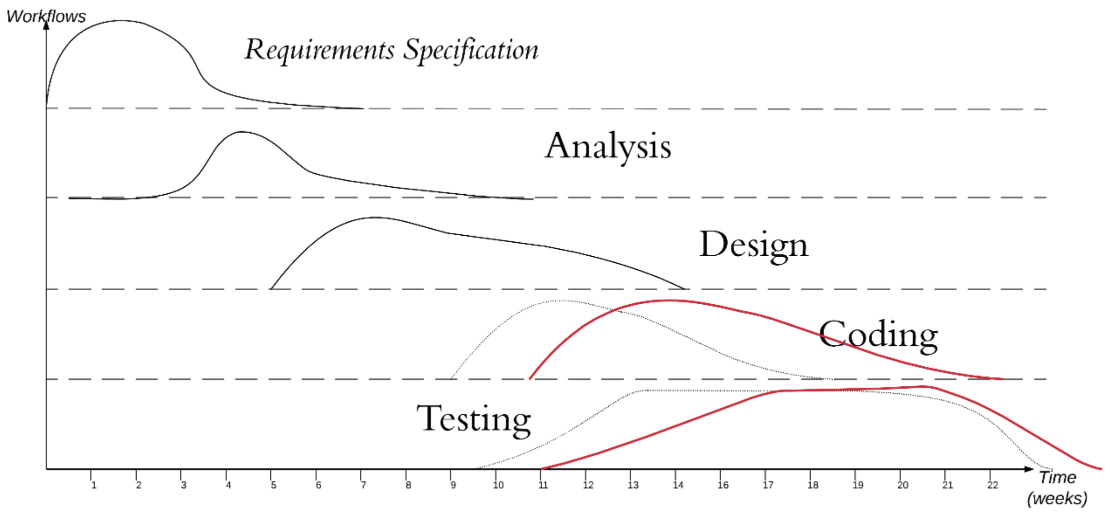
Conversely, if the analyst's capacity (ACAP) is better, his performance in the tasks of the sub-phase of analysis should be better, so his workload should start earlier. And as in the previous case, there is a chain effect with respect to the other sub-phases, which are also expected to start earlier, and as a general effect improve the execution time for a life-cycle of the development process. (See the following figure illustrating this effect)



Let us think of another example of this type of effects. What may it happen to the whole process if the quality of the programmers (PCAP) improves? As you may be thinking, in this case, probably the coding sub-phase would start earlier and would take a shorter time than in the nominal case. The drawn workflow for "Coding" and "Testing" have moved to the left and are narrower than in the nominal case, improving the general time that the whole life-cycle takes, thus, improving the whole process performance.



But there are more factors beyond the evident personnel attributes affecting the software development process. As an example, what may the effect of reliability (RELY) of the software used in the process of construction be? If the software platform is not reliable enough, wouldn't we expect the sub-phases of coding and testing to take longer and, thus, affect the whole life-cycle by adding more time to the total development time. See the following figure exemplifying the effect of software platform reliability (RELY) on the process.



At this moment, we can say that all of the factors may have an effect on the performance of the phases of a development cycle, and then in the whole process of software development. The increasing number of factors, the co-dependencies among these factors, and the emerging relationships among them, make of the software development process a complex process that cannot easily be explained and represented, so, it is also needed a complex approach, able to understand and represent such relations to appropriately address these issues.

[NOTE: Following sections (2, 3 and 4) are aimed to be discussed during our next session. To work with this, you will be using a web form that you will fill in, however you can read in the following lines a description about them.]

2) First assessment of comprehending questions.

You will have to reply to some questions regarding the performance or behavior of a software development process that follows an iterative development method like the one you already know.

Please provide answers in the first part of the form available here.

3) Video: "System Dynamics and the Software Process"

Now, you will be shown how the whole software development process that follows an iterative development method like the one you already know.can be represented through a System Dynamics simulation model and how such a process may evolve in accordance with the COCOMO II drivers (or factors) variation.

You will be shown a simulation of the Software Development Process Accomplishment and the effort expenditure behavior. You will be shown an attribute modification effect demonstration to illustrate the effect of changes in COCOMO II drivers over the whole process. Having into account that the previously mentioned factors are critical for such accomplishment, in section 4 **you will have to suggest how the behavior of the phases and the effort expenditure curve are affected after some specific factors or drivers are modified in a specific way**.

Play the video, available here-

4) Second assessment of comprehending questions.

You will have to reply to some questions regarding the performance or behavior of a software development process that follows an iterative development method like the one you already know.

Please provide answers in the second part of the form available here.

References

Boehm, B. (2012). *COCOMO II Model Definition Manual*. *87*(5 Suppl), i. https://doi.org/10.4269/ajtmh.2012.875suppack

Dugarte-Peña, G.-L. (2016). Software Engineering under the prism of System Dynamics. *XXIII Jornadas. Internacionales de Ingeniería de Sistemas. Universidad Católica de Santa María.*

Dugarte-Peña, G.-L., Sanchez-Segura, M.-I., Medina-Dominguez, F., & Amescua, A. de. (2020). Simulation of the software development process: an approximation using System Dynamics and the Larman Method / Simulación del proceso de desarrollo de software: una aproximación con Dinámica de Sistemas y el Método de Larman. *Revista Innovación y Software*, *1*(1), 39–57. https://revistas.ulasalle.edu.pe/innosoft/article/view/11

Larman, C. (2015). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd ed.). Prentice Hall.

Sanchez-Segura, M.-I. (2016). Software Economics under the prism of Systems Thinking. *XXIII Jornadas. Internacionales de Ingeniería de Sistemas. Universidad Católica de Santa María.*

Wikipedia. (2019). *Unified Process*. https://en.wikipedia.org/wiki/Unified_Process#Construction_phase