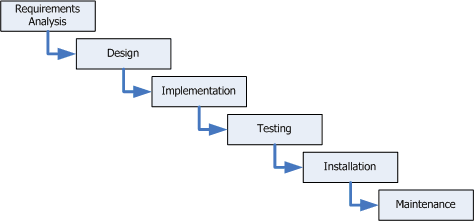
**2.0 METHODOLOGY OF THE STUDY**

The Waterfall Model will be used in the system development life cycle to create the Faculty Evaluation System of GCIC students. It is termed as waterfall because the model develops systematically from one phase to another in a downward fashion. This model is divided into different phases and the output of one phase is used as the input of the next phase. Every phase has to be completed before the next phase starts and there is no overlapping of the phases. In this methodology, the requirement of the client is captured and all the necessary documentation is done

The Waterfall method makes the assumption that all requirements can be gathered up front during the Requirements phase (Kee, 2006). Communication with the user is front-loaded into this phase, as the Project Manager does his or her best to get a detailed understanding of the user's requirements. Once this stage is complete, the process runs "downhill" (Hoffer, et al, 2008). Between each of those phases there are often “gates” that consist of reviews of produced material. In order to pass those “gates”, review of produced material needs to confirm that the previous phase is indeed complete. It usually ends with some form of a sign off.

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. This model very simple to understand and use. It is easy to manage as each phase has specific outputs and review process, it has clearly-defined stages

The students are confident to use this model to ensure success of their project since it is tested to work well for smaller projects where requirements are very clear. Process and output of each phase are clearly mentioned in the document.

[](https://www.google.com.ph/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiW35Onx7vWAhUIW7wKHboiBasQjRwIBw&url=http://nealcabage.com/waterfall-or-iterative-methodology/&psig=AFQjCNF3ggeKj9cmqMZZstChJ7dbEtHbig&ust=1506264649331518) Figure 2-1 Waterfall Method (SDLC)

**2.1 System Engineering and Analysis**

Because software almost always forms part of a much larger system, work begins by establishing requirements for all components of the system, identifying not only the role played by the software but, more importantly, its interface and interaction with the outside world. The students, in this phase, will carefully consider how the software will work when interfaced with other elements such as hardware, people, databases and computers too.

System engineering in the waterfall model must determine the factors that can influence, or restrict the flow of the system. All the possible issues must be explored, like for instance, the students must be sure that the software will fit into the whole Faculty Evaluation System. The software must have the overall picture, it must determine what the system will do exactly which in the case of this project, it should generate reports, tallies and summaries of the result. Even the environment where the system will be placed in, should be carefully considered. Will it be in the faculty office, will it be mobile based?, will the system be ‘real time’, ‘on-line’, ‘batch’, ‘safety critical’, ‘fault tolerant?’

It is actually important that the proposed system entitled “Web-based Faculty Evaluation System” is designed to develop an existing system because the system engineer in this part can assess the software solution to the existing problem of the system in place. Once all the mentioned factors, in the least, has been carefully analyzed and studied, the students will now be in a good position to assess the RISK involved in implementing the system. Once the systems engineering and analysis phase has been completed, and a picture of the role the software plays in the overall system has been established, the analysis can now focus specifically on the software and its requirements.

**2.2 Software Requirement**

Requirements Analysis is the 1st essential step towards creating a specification and a design. During this initial phase, the potential requirements of the application are methodically analyzed and written down in a specification document that serves as the basis for all future development. The result is typically a requirements document that defines what the application should do, but not how it should be done.

In the requirement analysis, the students and the faculty can work together to uncover the need of the faculty and asses how the software solution will have the ability to work successfully. The students must be able to pinpoint in this stage, the software’s function and performance. It should be determined how fast will the system respond based on the amount of data loaded. The result obtained after the test data has been used will also assess the correctness of the software

The students should also consider the fact that, when working with the faculty in terms of their expected result, they will be focused only on the problems from the user’s point of view. It is understandable that they may not have enough understanding of computers or software and may not even have a complete picture of how the system will behave at the initial stages of analysis. The objective of requirements analysis then, is to create a “requirements specification document” or a “target document” that describes in as much detail and in a very clear manner as possible exactly what the product should do. This requirements document will then form the basis of the subsequent design phase

**2.2.1 Design**

Once the analysis of the system has been completed, design or development can begin. This is an attempt to translate a set of requirements and program/data models that were laid down in the “requirements document” into a well-designed and engineering software solution. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture. It means that the design phase basically has subphases. There is a Logical Design subphase where the students will make use of the information collected so they can design the system without being dependent to any hardware or software. Once it is completed, then the students can now transform it to Physical design which is dependent to specific hardware and software. It is important in this phase that the functions and operations are described in detail including layouts, diagrams and other documentation. Once the preliminary requirements has been identified, a set of one or more design elements will be produced and developed as a result of interviews, workshops. All these design elements will now describe the software in sufficient detail that the students may develop the software with minimal additional input design.

**2.2.2 Coding**

In the Coding part, the student will now implement the modules in the design. The specification and model fed into the design stage can now be translated into code. The description of the roles or functions within the program is expanded and translated into an algorithm, which describes in detail exactly what, when and how the module/class carries out its task.

The modules algorithm can then be translated into a flowchart, which is a step-by-step graphical representation of the actions carried out by the module expressed in terms of sequence, selection and repetition. The flowchart can then be translated into Pseudocode, which conveys the same information as a flowchart, but presents it a way that is more amenable to translation into program code. Finally, the Pseudocode for each module is translated into a chosen programming language and the various modules entered, compiled, integrated into a system ready for testing. At each stage, the process is documented so that if changes are required in the future, the design pertinent to each stage is available for consultation and discussion.

**2.2.3 Testing**

Once the software components/modules have been written, testing can now begin. All the requirements stated or designs implemented will now be tested before they will be integrated in the system. Students can refer back to the scenarios presented in the implementation stage to be able to run the tests in an easy and smooth manner. In this stage, work is mostly based on quality checking (QC). The students should be very meticulous at this time because they have to prevent all errors and make sure that the system will be ready for deployment. The hardest part of this phase is when there is no full coordination with the system’s direct users which will mean that all the work in this phase is based on the Requirements document, validation can check whether the software conforms to the document, not whether it fulfills the user’s expectations. Unit, system and user acceptance testing’s should then be performed.

**2.2.4 Maintenance**

Once the Testing has been done and the system has been operational, the students will not stop from there. They will be expecting issues to come up when the system has been up and running in the faculty environment. GCIC students will commit into resolving these issues and they are expected to enhance the system whenever newer versions of the software is being released. Maintenance will be consistently done in the Faculty Evaluation System. Maintaining the system is an important aspect of The System Development Life Cycle (SDLC). Further, maintenance does not only take place when there is a software update, as key personnel change positions, or when an authorized user resigned and a new one came in the organization, new changes will be implemented, which will require system updates as well. Normally, maintenance will be handled by a dedicated Team, in this project’s case, the same students who handled the implementation and testing phase will take care of everything but most probably a manual of some sort will also be in place so that when the students will move out from the school, the system will still be maintained and enhanced by their successors.