INTRODUCTION

In a school, there are many departments that are encountering the same problem. The problem they facing are the management of information of every student including the following: personal information, the status of every subject they have taken, the list of prerequisite subject needs to be taken, the outstanding balance of fees.

The problem that existing in the Department of Computer Studies, the student’s information file and status of curriculum (subject completion) in a department are not complete and/or out-of-date. The causes are: There’s no one who manage student’s information and documentation so the department doesn’t have information about the student / files are incomplete; There’s no record of subject status so the student cannot determine if the subject needs to be taken or not; The student doesn’t know when will they update their information and they do not know what requirements to accomplish so the student may not update their information on time.

We are developing an Online Student’s Information Record System to help the Computer Studies department to organize information of students under in theirs. The purpose of the system is to give ease the student from updating their personal information, viewing the status of subject they have taken, viewing what are the prerequisite subject needs to be taken.

REVIEW OF RELATED LITERATURE

*What is System Development?*

According to Isaias P.; Issa, T. (2015)*System Development* consists of this process of creating

an information system, with all the variables that it entails and which usually

need to be taken into account: its ability to be user-friendly, how well it functions, if

it meets the needs of the organization in which it will be integrated, and so forth.

According to Dennis, Wixom & Roth (2012) S*ystem development* projects follow essentially the same fundamental process called the system development life cycle (SDLC). The SDLC starts with a planning phase in which the project team identifies the business value of the system, conducts a feasibility analysis, and plans the project. The second phase is the analysis phase,

in which the team develops an analysis strategy, gathers information, and builds a

set of analysis models. In the next phase, the design phase, the team develops the

design strategy, the physical design, architecture design, interface design, database

and file specifications, and program design. In the final phase, implementation, the

system is built, installed, and maintained.

According to Sommerville, Ian (2011) *System development* process are to develop or acquire all of the components of a system and then to integrate these components to create the final

system. The requirements are the bridge between the procurement and the development

processes. During procurement, business and high-level functional and nonfunctional

system requirements are defined.

**The Waterfall**

According to Isaias, P.; Issa, T. (2015) The waterfall model was introduced by Royce in (1970), specifically in the context of spacecraft mission software design, and is one of the most popular methods of assessing the evolution of a product or system. Essentially, it is a step-by-step sequential description of the product’s life cycle that spans 7 different stages, originally denominated “system requirements, software requirements, analysis, program design, coding, testing and operations” (Royce 1970). The waterfall model was a popular approach, and for that reason, it evolved and adapted into numerous forms, according to different research studies and the context of application. Denomination of each step varies greatly and can reflect the specific objective of the study or the field in which it is applied.

According to Dennis, Wixom & Roth (2012) analysts and users proceed sequentially from one phase to the next. The key deliverables for each phase are typically voluminous (often, hundreds of pages) and are presented to the approval committee and project sponsor for approval as the project moves from phase to phase.

According to Sommerville, Ian (2011) Because of the cascade from one phase to another, this model is known as the ‘waterfall model’ or software life cycle. The waterfall model is an example of a plan-driven process—in principle, you must plan and schedule all of the process

activities before starting work on them.

According to Marsic, Ivan (2012) The early inspiration for software lifecycle came from other engineering disciplines, where the above activities usually proceed in a sequential manner (or at least it was thought so). This method is known as *waterfall process* because developers build monolithic systems in one fell swoop. It requires completing the artifacts of the current phase before proceeding to the subsequent one. In civil engineering, this approach would translate to: finish all blueprints neatly before starting construction; finish the construction before testing it for soundness; etc. There is also psychological attraction of the waterfall model: it is a linear process that leads to a conclusion by following a defined sequence of steps.

*Steps*

According to Isaias P.; Issa, T. (2015) the essence of the waterfall model is that it attempts to provide a useful set of guidelines for the development of new programs or systems. In his original work, Royce (1970) provides five key principles that he believes are essential for the

successful development of large software systems. The first is “program design comes first.” It is essential to allow designers to be a part of the initial process, because of their invaluable feedback regarding resources and limitations. The second is “document the design.” Extensive documentation of the development process is paramount, not just to facilitate management of the process, but to facilitate performance assessments, making the eventual correction

of mistakes more efficient. The third is “do it twice,” referring that the final version

of the product should actually be the second version, where all the stages have been

performed and it is easier to pinpoint strengths and weaknesses, to emphasize the

first and correct the latter. The fourth is “plan, control and monitor testing.” Testing

is a fundamental stage. It is important to bring in specialists that did not participate

in the earlier stages of the process. It is also important to test every single aspect of

the project, regardless of how relevant it is. Finally, the fifth guideline is “involve

the customer.” Having the insight, judgment, and commitment of the customer

taken into account during the development process is a viable option that will

greatly improve its potential for general acceptance.

According to Dennis, Wixom & Roth (2012) Once the work produced in one phase is approved, the phase ends and the next phase begins. As the project progresses from phase to phase, it

moves forward in the same manner as a waterfall. While it is possible to go backward

through the phases (e.g., from design back to analysis), it is quite difficult.

According to Sommerville, Ian (2011) The principal stages of the waterfall model directly reflect the fundamental development activities:

1. ***Requirements analysis and definition***

The system’s services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.

1. ***System and software design***

The systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.

1. ***Implementation and unit testing***

During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.

1. ***Integration and system testing***

The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.

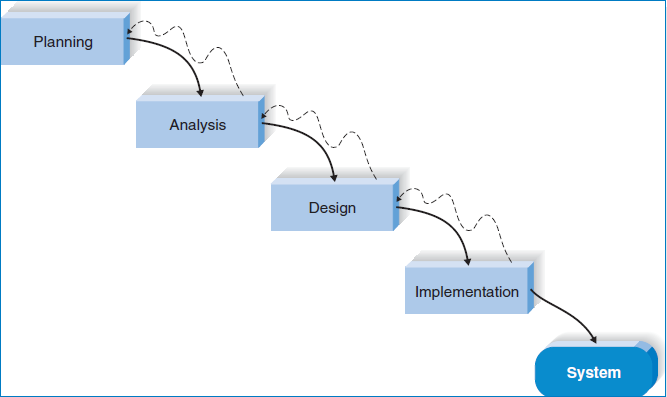
1. ***Operation and maintenance***

Normally (although not necessarily), this is the longest life cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system’s services as new requirements are discovered.

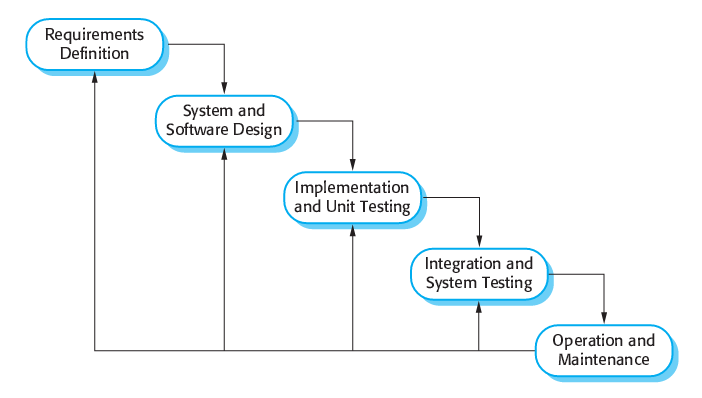
*Diagram*



**Fig. 1** Waterfall life cycle model by to Isaias P.; Issa, T.



**Fig 1.2** Waterfall life cycle model by Isaias P.; Issa, T.



**Fig 1.3** Waterfall life cycle model by Sommerville, Ian

*Advantages*

According to Isaias P.; Issa, T. (2015) it is easy to understand and easy to use; provides structure to inexperienced staff; sets requirements stability and good for management control.

According to Dennis, Wixom & Roth (2012) Waterfall development methodologies have the advantages of identifying requirements long before programming begins and limiting changes to the requirements as the project proceeds.

According to Sommerville, Ian (2011) The waterfall model is consistent with other engineering process models and documentation is produced at each phase. This makes the process visible so managers can monitor progress against the development plan.

*Disadvantages*

According to Isaias P.; Issa, T. (2015) all requirements must be known upfront; system can be frozen before the design begins and little opportunity for customer to preview the system.

According to Dennis, Wixom & Roth (2012) The key disadvantages are that the design must be

completely specified before programming begins, a long time elapses between the

completion of the system proposal in the analysis phase and the delivery of system,

and testing is treated almost as an afterthought in the implementation phase. If the project team misses an important requirement, expensive post-implementation programming may be needed. Users may forget the original purpose of the system, since so much time has elapsed between the original idea and actual implementation. Also, in today’s dynamic business environment, a system that met the existing environmental conditions during the analysis phase may need considerable rework to match the environment when it is implemented. This rework requires going back to the initial phase and making needed changes through each of the subsequent phases in turn. In addition, the deliverables are often a poor communication mechanism, so important requirements may be overlooked in the volumes of documentation.

According to Sommerville, Ian (2011) it’s major problem is the inflexible partitioning of the project into distinct stages. Commitments must be made at an early stage in the process, which makes it difficult to respond to changing customer requirements. In principle, the waterfall model should only be used when the requirements are well understood and unlikely to change radically during system development. However, the waterfall model reflects the type of process used in other engineering projects. As is easier to use a common management model for the whole project, software processes based on the waterfall model are still commonly used.

**Prototyping Model**

According to Isaias P.; Issa, T. (2015) The *prototyping model* is an iterative framework that is at the center of many of the more agile approaches to software development, ever since the early 1980s, which lead to it being described in some studies as a specific model in itself. In 1997, Carr and Verner observed that in the past research, the SDLC models that adopted

prototyping were found to be more dynamic and more responsive to client needs, as

well as less risky and more efficient. For that reason, they attempted to summarize

prototyping models in one consistent framework. The prototyping model is based on the idea of creating the entirety or part of a system in a pilot version, called the prototype. It can be viewed as a process, either one that is part of the larger SDLC or the central approach that defines the SDLC in itself. The goal is ultimately to build in various versions and consistently refine those versions until a final product is reached.

According to Dennis, Wixom & Roth (2012) *prototyping* *model* performs the analysis, design, and implementation phases concurrently in order to quickly develop a simplified version of the proposed system and give it to the users for evaluation and feedback.

According to Sommerville, Ian (2011) A prototype is an initial version of a software system that is used to demonstrate concepts, try out design options, and find out more about the problem and its possible solutions. Rapid, iterative development of the prototype is essential so that costs are controlled and system stakeholders can experiment with the prototype early in the software process.

*Steps*

According to Isaias P.; Issa, T. (2015) A *prototyping model* essentially entails four different stages. First, user’s requirements and needs are analyzed and identified. Next, the team will develop a working prototype of the product, which is then implemented so that the users can

test it and provide real-time feedback and experience. If improvements and changes

are found necessary, the prototype is revised and refined, and a new prototype is

released and implemented for testing. This subcycle will go on until the product is

generally accepted by the users and no longer requires substantial changes or

updates, at which time the final version is released.

According to Dennis, Wixom & Roth (2012) This cycle continues until the analysts, users, and sponsor agree that the prototype provides enough functionality to be installed and used in the

organization. System prototyping very quickly provides a system for users to evaluate

and reassures users that progress is being made.

According to Sommerville, Ian (2011) The objectives of prototyping should be made explicit from the start of the process. These may be to develop a system to prototype the user interface, to develop a system to validate functional system requirements, or to develop a system to demonstrate the feasibility of the application to managers. The same prototype cannot meet all objectives. If the objectives are left unstated, management or end-users may misunderstand the function of the prototype. Consequently, they may not get the benefits that they expected from the prototype development. The next stage in the process is to decide what to put into and, perhaps more importantly, what to leave out of the prototype system. To reduce prototyping costs and accelerate the delivery schedule, you may leave some functionality out of the prototype. You may decide to relax non-functional requirements such as response time and memory utilization. Error handling and management may be ignored unless the objective of the prototype is to establish a user interface. Standards of reliability

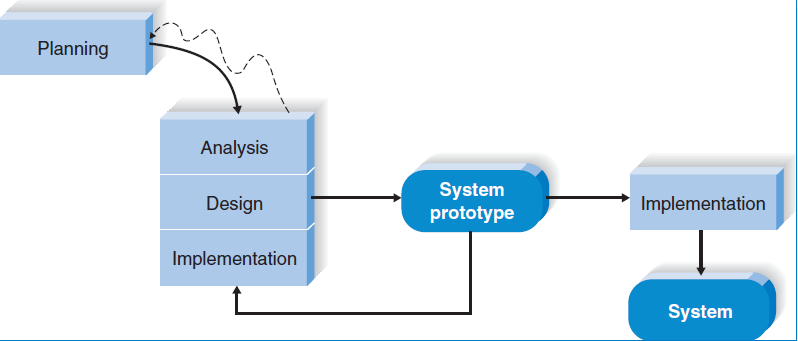
and program quality may be reduced. The final stage of the process is prototype evaluation. Provision must be made during this stage for user training and the prototype objectives should be used to derive a plan for evaluation. Users need time to become comfortable with a new system and to settle into a normal pattern of usage. Once they are using the system normally,

they then discover requirements errors and omissions.

*Diagram*



**Fig. 2** Prototyping model by Isaias P.; Issa, T.



**Fig. 2.1** Prototyping model by Dennis, Wixom & Roth

*Advantage*

According to Isaias P.; Issa, T. (2015) By using a form of the prototyping model, a development project can easily adapt to changing requirements, because there is constant feedback. With each iteration, or version of the product, the user will have the ability to test the prototype

and provide valuable input on its traits and requirements. This provides the model

with much higher probabilities of success, as well as low risks. On the other hand,

because there is not much emphasis on extensive documentation, and the product

evolves as it is created, the time frame for the development project is much shorter

than with rigid models.

According to Dennis, Wixom & Roth (2012) The approach is very useful when users have difficulty expressing requirements for the system.

*Disadvantages*

According to Isaias P.; Issa, T. (2015) However, prototyping models are weak on analysis and design planning. While requirements are assessed as the product is developed in successive versions, there is little control over costs and resources, which can dramatically increase the

financial cost of the project.

According to Dennis, Wixom & Roth (2012) However is the lack of careful, methodical analysis prior to making design and implementation decisions. System prototypes may have some fundamental design limitations that are a direct result of an inadequate understanding of the system’s true requirements early in the project.

According to Sommerville, Ian (2011) a general problem with prototyping is that the prototype may not necessarily be used in the same way as the final system. The tester of the prototype may not be typical of system users. The training time during prototype evaluation may be insufficient. If the prototype is slow, the evaluators may adjust their way of working and avoid those system features that have slow response times. When provided with better response in the final system, they may use it in a different way.

**Rapid Application Development**

According to Isaias P.; Issa, T. (2015) it is driven by the idea that existing life cycle models are simply too rigid to permit a fast project development; therefore, there is need for a

framework that can account for fast delivery while still maintaining high-quality

standards. It is grounded on the principle that step-by-step structured life cycles

inevitably entail delays and errors, urging the need for an alternative methodology.

*Steps*

According to Isaias P.; Issa, T. (2015) it is possible to analyze requirements, alternatives, andopportunities, as well as possible risks. This will form the basis for a definition of the project’s goals and scope, and more importantly, it will allow for the establishment

of the timebox, which is a fixed period during which a specific increment of the product is going to be developed. Each increment is then developed in a spiral-like model, through design, prototyping, and testing. This method essentially pushes the team closer to the project’s business goals, by providing key deadlines that can be determined by market forces.

According to Dennis, Wixom & Roth (2012) *Rapid application development* is a collection

of methodologies that emerged in response to the weaknesses of waterfall development and its variations. RAD incorporates special techniques and computer tools to speed up the analysis, design, and implementation phases in order to get some portion of the system developed quickly and into the hands of the users for evaluation and feedback. CASE (computer-aided software engineering) tools, JAD (joint application development) sessions, fourth-generation/visual programming languages (e.g., Visual Basic.NET), and code generators may all play a role in RAD.

*Diagram*



*Advantages*

According to Isaias P.; Issa, T. (2015) The advantages of the RAD are evident, due to its focus on swift delivery and effective developer–client communication.

According to Dennis, Wixom & Roth (2012) *RAD* can improve the speed and quality of systems development, it may also introduce a problem in managing user expectations. As systems are developed more quickly and users gain a better understanding of information technology, user expectations may dramatically increase and system requirements may expand

during the project.

*Disadvantages*

According to Isaias P.; Issa, T. (2015) There are still a number of issues raised by this approach. One of the most obvious flaws is that it removes a great deal of emphasis on minute planning and modeling at the start of the project, shifting that focus to the fluid process of system construction. Another prominent issue is that in faster development cycles, extensive quality

testing will become less prioritized, reflecting in poorer quality overall, which means that effective RAD methodologies should reserve space for skilled individuals in quality control roles It is also possible that managers and leader have unrealistic expectations regarding the timeboxes, creating conflict with developing teams.

According to Dennis, Wixom & Roth (2012) it is possible to assert that in order to be optimized, RAD life cycles must necessarily be balanced and be open to moderating agents.

**The V Life Cycle Model**

According to Isaias P.; Issa, T. (2015) the V-Model was presented in the final years of the 1980s by Paul Rook, as a variation over the waterfall model that attempted to emphasize the existing connection between each of the stages of the development process and it’s respective

stage of tests. By focusing on this relationship, it ensures that adequate quality

measurements and testing are constantly resorted to throughout the life cycle.

According to Dennis, Wixom & Roth (2012) The *V-model* is another variation of waterfall development that pays more explicit attention to testing. A key concept of this model is that as requirements are specified and components designed, testing for those elements is also defined. In this manner, each level of testing is clearly linked to a part of the analysis or design phase, helping to ensure high quality and relevant testing and maximize test effectiveness.

*Steps*

According to Isaias P.; Issa, T. (2015) the V-Model starts off with a very similar premise to the classic waterfall models. In successive steps, the project goes from analysis of requirements and

specifications, to architectural and detailed design, to coding. However, instead of

continuing this downward ladder, there is a parallel structure that moves upward

from the coding stage, giving the model its distinct V shape. The upward ladder

describes each of the testing steps that follows coding, starting with unit testing and

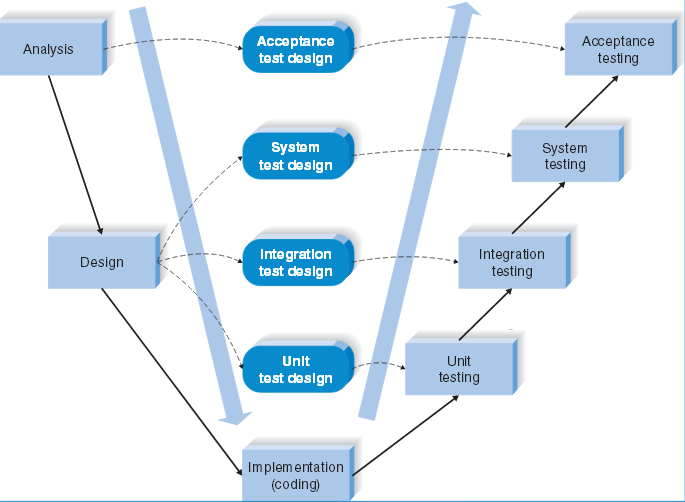
ending with acceptance testing, the final step before final release. In that sense, the V-Model describes three successive layers of system development that can be described as requirements (overall system), high-level design (system architecture), and low-level design (software components). To each of these layers, there is a corresponding layer of planning and testing. Planning is, indeed, the axis that stands between the left and right ladders that compose the V, as it is the mediating action set between design and testing.

According to Dennis, Wixom & Roth (2012) the development process proceeds down the left-hand slope of the V, defining requirements and designing system components. At the base of the V, the code is written. On the upward-sloping right side of the model, testing of components, integration testing, and, finally, acceptance testing are performed.

*Diagram*



**Fig. 4** V life cycle model by to Isaias P.; Issa, T.



**Fig. 4.2** V-Model by Dennis, Wixom & Roth

*Advantages*

According to Isaias P.; Issa, T. (2015) Since the V-Model addresses its errors shortly after they are identified; it becomes less expensive to resolve them, which is perhaps the greatest advantage of using this model, specifically when compared to the classic waterfall model. Also,

because testing is fractioned throughout the process, all the parties of the development

are responsible for it. This also means that testing methods are adequate to

each of the stages. Furthermore, the fact that tests are performed since the beginning

of the process only increases its efficiency.

According to Dennis, Wixom & Roth (2012) The V-model is simple and straightforward and improves the overall quality of systems through its emphasis on early development of test plans. Testing focus and expertise is involved in the project earlier rather than later; plus, the testers gain knowledge of the project early.

*Disadvantages*

According to Isaias P.; Issa, T. (2015) this model is very rigid and there is little room for flexible adaptation, particularly because any alteration in the requirements will render all existing documentation and testing obsolete. Since it requires a great deal of resources, it is clearly optimized for large projects within large organizations.

According to Dennis, Wixom & Roth (2012) It still suffers from the rigidity of the waterfall

development process, however, and is not always appropriate for the dynamic

nature of the business environment.

**Agile**

According to Isaias P.; Issa, T. (2015) With the popularization of waterfall-like SDLC models, an alternative approach has been developing that attempts to counter their rigidness and lack of flexibility. We have seen such examples in the incremental and RAD models. In 2001, the manifesto for agile software development was presented by 17 software developers, in a

new attempt to bring together the best traits of other agile-like models into one

framework. Since then, agile methods of development have become increasingly

popular.

According to Dennis, Wixom & Roth (2012) *Agile development* is a group of programming-centric methodologies that focus on streamlining the SDLC. Much of the modeling and documentation overhead is eliminated; instead, face-to-face communication is preferred. A

project emphasizes simple, iterative application development in which every iteration is a complete software project, including planning, requirements analysis, design, coding, testing, and documentation. Cycles are kept short (one to four weeks), and the development team focuses on adapting to the current business environment. There are several popular approaches to agile development, including extreme programming, Scrum, and dynamic systems development method (DSDM).

According to Sommerville, Ian (2011) *Agile methods* are incremental development methods in which the increments are small and, typically, new releases of the system are created and made available to customers every two or three weeks. They involve customers in the development process to get rapid feedback on changing requirements. They minimize documentation by using informal communications rather than formal meetings with written documents. Agile methods universally rely on an incremental approach to software specification, development, and deliver.

According to Marsic, Ivan (2012) Agility is both a development philosophy and a collection of concepts embedded into development methodologies. An agile approach to development is essentially a results-focused method that iteratively manages changes and risks. It also actively engages customers in providing feedback on successive implementations, in effect making them part of the development team. Unlike process-driven documentation, it promotes outcome-driven documentation. The emphasis of agile practices is on traveling lightweight, producing only those artifacts (documentation) that are absolutely necessary.

*Steps*

According to Isaias P.; Issa, T. (2015) There are numerous subvariations of the Agile model that follow these principles, with some examples being the scrum and XP models. However, even considering the variation in timescales or stage description, it is possible to determine

the general path that an agile development process will take, outlined in four steps.

The first step is *project selection and approval*. During this stage, a team consisting

of developers, managers, and customers establishes the scope, purpose, and

requirements of the product. There is also a thorough analysis of different alternatives to accomplish the established goals, as well as risk assessment for each

idea.

The second step is *project initiation*. After the establishment of a coherent project with respective goals and scope, a working team is built, with the appropriate environment and tools, as well as the working architecture in which the

system will be based. This too is discussed among all stakeholders. At this point, it

is also adequate to establish working time frames and schedules.

The third step is construction iterations, with each iteration consisting of both planning and building. Developers release working software in successive increments that will accommodate the evolution of requirements as outlined by the

various stakeholders. Close collaboration is therefore a fundamental aspect of this

process, as the most effective method to ensure quality and to keep the project’s

priorities well defined. Extensive testing of each iteration is also paramount at this

point.

The fourth and final step is *product release*. This stage encompasses two stages: First, final testing of the entire system is done, as well as any necessary final reworks and documentations. Next, the product is released, at which point training is provided to the users in order to maximize operational integration. The working

team might maintain the project so as to allow for product improvement as well as

user support.

According to Dennis, Wixom & Roth (2012) It begins with user stories that describe what the system needs to do. Then, programmers code in small, simple modules and test to meet those needs. Users are required to be available to clear up questions and issues as they arise. Standards are very important to minimize confusion, so XP teams use a common set of

names, descriptions, and coding practices. XP projects deliver results sooner than even

the RAD approaches, and they rarely get bogged down in gathering requirements for

the system.

*Diagram*



**Fig. 5** An example of an Agile development life cycle

*Advantages*

According to Isaias P.; Issa, T. (2015) The Agile SDLC emerged from the ever-increasing need to match the speed at which IT evolves. What sets it apart is its dexterity in developing products at a great speed, with products being deliverable in the course of weeks instead of months.

This is possible due to the model’s emphasis on collaborative efforts and documentation.

Another advantage of the Agile model is that it is very flexible. It has been

occasionally combined with other existing models. It has the capacity to deliver

systems whose requirements go through constant changes while, at the same time,

demanding strict time limits.

According to Dennis, Wixom & Roth (2012) Agile development approaches have

existed for over a decade. Agile development practices were created in part

because of dissatisfaction with the sequential, inflexible structure of waterfall-based

approaches. Presently, agile development has made inroads into software

development organizations, and studies show an even split between agile and

waterfall users. Many organizations are experimenting with agile even while

continuing to employ traditional waterfall approaches.

According to Sommerville, Ian (2011) They are best suited to application development where the system requirements usually change rapidly during the development process. They are intended to deliver working software quickly to customers, who can then propose new and changed requirements to be included in later iterations of the system. They aim to cut down on process bureaucracy by avoiding work that has dubious long-term value and eliminating documentation that will probably never be used.

*Disadvantages*

According to Isaias P.; Issa, T. (2015) The downside of this if in case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle. There is lack of emphasis on necessary designing and documentation. The project can easily get taken off track if the customer representative is not clear what final outcome that they want. Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

According to Dennis, Wixom & Roth (2012) Consequently, the use of XP in combination with outside contractors produces a highly questionable outcome, since the outside contractors may never “jell” with insiders. XP requires a great deal of discipline to prevent projects from becoming unfocused and chaotic. Furthermore, it is recommended only for small groups of developers (not more than 10), and it is not advised for mission-critical applications. Since little analysis and design documentation is produced with XP, there is only code documentation; therefore, maintenance of large systems developed using XP may be impossible. Also, since

mission-critical business information systems tend to exist for a long time, the utility

of XP as a business information system development methodology is in doubt. Finally,

the methodology requires considerable on-site user input, something that is frequently

difficult to obtain.

According to Sommerville, Ian (2011) However, when this heavyweight, plan-driven development approach is applied to small and medium-sized business systems, the overhead involved is so large that it dominates the software development process. More time is spent on how the system should be developed than on program development and

testing. As the system requirements change, rework is essential and, in principle at least, the specification and design has to change with the program.

Programming Languages

**PHP**

*PHP* is a programming language for building dynamic, interactive Web sites. As a general rule, PHP programs run on a Web server, and serve Web pages to visitors on request. One of the key features of PHP is that you can embed PHP code within HTML Web pages, making it very easy for you to create dynamic content quickly. PHP is a server - side scripting language, which means that PHP scripts, or programs, usually run on a Web server. (A good example of a

client – side scripting language is JavaScript, which commonly runs within a Web browser.) Furthermore, PHP is an interpreted language — a PHP script is processed by the PHP engine each time it’s run. We should use php because one of the best things about PHP is the large number of Internet service providers (ISPs) and Web hosting companies that support it. Today hundreds of thousands of developers are using PHP, and it’s not surprising that there are so many, considering that several million sites are reported to have PHP installed.

**Phyton**

Python is an uncomplicated and robust programming language that delivers both the power

and complexity of traditional compiled languages along with the ease-of-use (and then some)

of simpler scripting and interpreted languages. You'll be amazed at how quickly you'll pick

up the language as well as what kind of things you can do with Python, not to mention the

things that have already been done. We should use python because it is an Internet and systems programming language that is soaring in popularity in today's fast-paced software development environment, and no wonder: it's simple (yet robust), object-oriented (yet can be used as a procedural language), extensible, scalable and features an easy to learn syntax that is clear and concise. Python combines the power of a compiled object language like Java and C++ with the ease of use and rapid development time of a scripting language. In fact, its syntax is so easy to understand that you are more likely to pick it up faster than any of the other popular scripting languages in use today. Although practically a decade in age, Python is still somewhat relatively new to the general software development industry. We should, however, use caution with our use of the word "relatively," as a few years seem like decades when developing on "Internet time."

**Ruby**

Ruby is a dynamic programming language with a complex but expressive grammar and a core class library with a rich and powerful API. Ruby draws inspiration from Lisp, Smalltalk, and Perl, but uses a grammar that is easy for C and Java™ programmers to learn. Ruby is a pure object-oriented language, but it is also suitable for procedural and functional programming styles. It includes powerful meta-programming capabilities and can be used to create domain-specific languages or DSLs.

**Javascript**

JavaScript was introduced in 1995 as a way to add programs to web pages in the Netscape Navigator browser. The language has since been adopted by all other major graphical web browsers. It has made modern web applications possible—applications with which you can interact directly, without doing a page reload for every action. But it is also used in more traditional websites to provide various forms of interactivity and cleverness. It is important to note that JavaScript has almost nothing to do with the programming language named Java. The similar name was inspired by marketing considerations, rather than good judgment. When

JavaScript was being introduced, the Java language was being heavily marketed and was gaining popularity. Someone thought it was a good idea to try to ride along on this success.

**Java**

Java is related to C++, which is a direct descendant of C. Much of the character of Java

is inherited from these two languages. From C, Java derives its syntax. Many of Java’s

object-oriented features were influenced by C++. In fact, several of Java’s defining

characteristics come from—or are responses to—its predecessors. Moreover, the creation of

Java was deeply rooted in the process of refinement and adaptation that has been occurring

in computer programming languages for the past several decades. For these reasons, this

section reviews the sequence of events and forces that led to Java. As you will see, each

innovation in language design was driven by the need to solve a fundamental problem

that the preceding languages could not solve. Java is no exception.

IDE

**Azure**

Azure Websites is a managed cloud service that allows you to deploy a web application and make it available to your customers on the Internet in a very short amount of time. You don’t directly support the VMs on which your website runs; they are managed for you. Supported languages include .NET, Java, PHP, Node.js, and Python. In addition to creating your own website, there are several web applications available to use as a starting point, such as WordPress, Umbraco, Joomla!, and Drupal. You can use continuous deployment with Team Foundation Server (TFS), Git, or GitHub so that every time you commit a change, a new version of the website is deployed. You have the ability to scale the number of instances in and out on demand; you also can configure autoscaling so Azure will scale it in or out for you

depending on specific performance measures such as CPU Percentage. If your website has multiple instances, you can configure load balancing to make the most of your resources.

**NetBeans IDE**

NetBeans is a Java Integrated Development Environment, IDE, which enables fast application development with the most adopted frameworks, technologies, and servers.

Different than other IDEs, NetBeans comes already pre-packaged with a wide range of

functionality out of the box, such as support for different frameworks, servers, databases,

and mobile development.

Hardware (Components)

**RFID**

RFID is an acronym for radio frequency identification, which is a wireless communication technology that is used to uniquely identify tagged objects or people. It has many applications. Some present-day examples include:

***THE THREE CORE COMPONENTS OF AN RFID SYSTEM***

An RFID system uses wireless radio communication technology to uniquely

identify tagged objects or people. There are three basic components to an

RFID system

1. A tag (sometimes called a transponder), which is composed of a semi-conductor chip, an antenna, and sometimes a battery

2. An interrogator (sometimes called a reader or a read/write device), which is composed of an antenna, an RF electronics module, and a control electronics module

3. A controller (sometimes called a host), which most often takes the form of a PC or a workstation running database and control (often called

middleware) software.

Database

**Microsoft Access**

On entering Access you are presented with screen showing available templates, which have been designed to help you create your own databases. You can search for further templates at Office.com. These templates can be quite helpful for particular applications but you nearly always have to tailor the database produced to your own requirements. You can also either create a new blank database (without help) or open an existing one. In this course you are going to use an existing database, to see how it is set up and how it can be used.

**MySQL**

The MySQL development process focuses on offering a very efficient implementation of the features most people need. This means that MySQL still has fewer features that its chief open source competitor, PostgreSQL, or the commercial database engines. Nevertheless, the skills you get from this book will serve you well on any platform. Many database management systems—even open source ones—preceded MySQL. Why has MySQL been the choice for so many beginners and small sites, and now for some heavyweight database users in government and industry? We can suggest a few

factors:

MySQL can run on very modest hardware and puts very little strain on system

resources; many small users serve up information to their organizations by running MySQL on modest desktop systems. The speed with which it can retrieve information has made it a longstanding favorite of web administrators. Over the past few years, MySQL AB has addressed the need of larger sites by adding features that necessarily slow down retrieval, but its modular design lets you ignore the advanced features and maintain the suppleness and speed for which MySQL is famous.

Partly because MySQL is small and fast, it works the way most people want straight “out of the box.” It can be installed without a lot of difficult and sophisticated configuration. Now that many Linux distributions include MySQL, installation can be almost automatic. This doesn’t mean MySQL is free of administrative tasks. In particular, we’ll cover a few things you need to do at the start to tighten security. Very little configuration is shown in this book, however, which is a tribute to the database engine’s convenience and natural qualities.

As we’ll explain in the “Structured Query Language” section later in this chapter, multiple standards exist in the relational database world, and it’s impossible to claim total conformance. But learning MySQL certainly prepares you for moving to other database engines. Moving code from one database engine to another is never trivial, but MySQL does a reasonable job of providing a standard environment, and gets better as it develops more features.

With a few hundred employees scattered around the globe, MySQL AB is a very

flexible organization that keeps constant tabs on user needs. At its conferences,

lead developers get out in front and make themselves available to everyone with a

gripe or a new idea. There are also local MySQL user groups in almost every major

city. This responsiveness is helped by the fact that MySQL is open and free; any

sufficiently skilled programmer can look at the program code to find and perhaps

help in fixing problems. MySQL actually has a dual-license approach: if you want to build your own product around it, you pay MySQL AB a license fee. If you just want to use MySQL to serve your own data, you don’t have to pay the license fee. MySQL also offers

technical support, as do numerous other companies and consultants, some of them

probably near you.

It is easy to use MySQL as part of a larger software system. For example, you can

write programs that can interact directly with a MySQL database. Most major

programming languages have libraries of functions for use with MySQL; these in-

clude C, PHP, Perl, Python, Ruby, and the Microsoft .NET languages. MySQL also

supports the Open Database Connectivity (ODBC) standard, making it accessible

even when MySQL-specific functionality isn’t available.

**Oracle Database**

An Oracle database is a collection of data in one or more files. The database contains physical and logical structures. In the course of developing an application, you create structures such as tables and indexes to store rows and speed their retrieval. You can create synonyms for the object names, view objects in different databases (across database links), and restrict access to the objects. You can even use external tables to access files outside the database as if the rows in the files were rows in tables. In this book, you will see how to create these objects and develop applications based on them. An Oracle

Instance comprises a memory area called the System Global Area (SGA) and the

background processes that interact between the SGA and the database files on disk. In an Oracle Real Application Cluster (RAC), more than one instance will use the same database. The instances generally are on separate servers connected by a high-speed interconnect.

Inside the Database

Within the Oracle database, the basic structure is a table. Oracle Database 11g supports many types of tables, including the following:

**Relational tables**

Using the Oracle-supplied, you can create tables to store the rows inserted and manipulated by your applications. Tables have column definitions, and you can add or drop columns as the application requirements change. Tables are created via the create table command.

**Object-relational tables**

To take advantage of features such as type inheritance, you can

use Oracle’s object-relational capabilities. You can define your own datatypes and then

use them as the basis for column definitions, object tables, nested tables, varying arrays,

and more.

**Index-organized tables**

You can create a table that stores its data within an index structure, allowing the data to be sorted within the table.

**External tables**

Data stored in flat files may be treated as a table that users can query

directly and join to other tables in queries. You can use external tables to access large

volumes of data without ever loading them into your database. See Chapter 28. Note that

Oracle also supports BFILE datatypes, a pointer to an external binary file. Before creating a

BFILE or an external table, you must create a directory alias within Oracle (via the

create directory command) pointing to the physical location of the file. See Chapter 40

for details on BFILEs and other large object datatypes.

**Partitioned tables**

You can divide a table into multiple partitions, which allows you

to separately manage each part of the table. You can add new partitions to a table, split

existing partitions, and administer a partition apart from the other partitions of the table.

Partitioning may simplify or improve the performance of maintenance activities and user

queries. You can partition tables on ranges of values, on lists of values, on hashes of

column values, or on combinations of those options.

**Materialized views**

A materialized view is a replica of data retrieved by a query. User queries may be redirected to the materialized views to avoid large tables during execution—the optimizer will rewrite the queries automatically. You can establish and manage refresh schedules to keep the data in the materialized views fresh enough for the business needs.

**Temporary tables**

You can use the create global temporary table command to create a table in which multiple users can insert rows. Each user sees only his or her rows in the table.

**Clustered tables**

If two tables are commonly queried together, you can physically store

them together via a structure called a cluster.

**Dropped tables**

You can quickly recover dropped tables via the flashback table to before drop

command. You can flash back multiple tables at once or flash back the

entire database to a prior point in time. Oracle supports flashback queries, which return

earlier versions of rows from an existing table.

Other Technologies

**Web Hosting**

In order to have a web site on the Internet, you need a place to store it so that the

public can find it. When you purchase web hosting, you are sort of renting space on the Internet. Actually, you are just renting hard drive space along with some very

useful applications and important website tools from a company that provides space

on the Internet. That “space” is called a web server and is provided by a web HOST.

The Web Host makes sure your website is available to people who are browsing the

Internet. You are also paying for bandwidth which, in basic terms, is just a

measurement of accessibility between the public and your website files. When

someone views your website, for example, they are spending your bandwidth.

A web host is a service provider that allows you to place your website on a computer (server) which is connected to the Internet. This gives people who surf the Internet a way to

access your website. The computer that the web hosting company uses is similar to the

computer you have at home, the difference being that it is set up to “serve” websites

and is therefore called a "server". It is a much more robust computer than the desktop pc at home and is usually loaded with countless software applications that make managing websites much easier. A web server also holds applications that send

and receive email, manage databases, work with web browsers, filter spam, install website tools, run website programs and scripts and much more.

A web host will typically have a fast connection to the Internet and they may host

thousands of web sites on many servers. The web host essentially rents out space to

you so that you can get your website up on the World Wide Web.

Web servers are usually stored in Data Centers – also known as Network Operations

Centers (NOC). These NOCs contains many servers all connected

to the Internet. It is very expensive to own and operate a datacenter. Datacenters have reliable and fast backbone connections to the Internet through several hubs and also have

backup power systems, clean room atmospheres and very high

security systems. Most web hosting companies do not own a datacenter, but rather co-locate their servers remotely into a professional datacenter.

Black Wire Hosting’s web servers are stored in more than one datacenter (or NOC)

for security purposes and for proper balance of our network of servers. We do this to

provide a wider range of connectivity and reliability for our hosting customers.

There are literally thousands of web hosting companies in the marketplace, all with

different types of plans and at different prices. It can be hard to wade through all the

options, especially if you are completely new to web hosting and just want to get

your website up on the web. All web hosting plans have some basic common feature.

Design

**Bootstrap**

We should use bootstrap because it is Mobile First Approach, Bootstrap 3 framework consists of Mobile first styles throughout the entire library instead of them in separate files; Browser Support, It is supported by all popular browsers; Easy to get started, with just the knowledge of HTML and CSS anyone can get started with Bootstrap. Also the Bootstrap official site has a good documentation; Responsive Design, bootstrap's responsive CSS adjusts to Desktops, Tablets and Mobiles. More about the responsive design is in the chapter Bootstrap Responsive Design. Bootstrap provides a clean and uniform solution for building an interface for developers. It contains beautiful and functional built-in components which are easy to customize. It also provides web based customization. And best of all it is an open source. Bootstrap includes Scaffolding, Bootstrap provides a basic structure with Grid System, link styles, and background. This is covered in detail in the section Bootstrap Basic Structure CSS, Bootstrap comes with the feature of global CSS settings, fundamental HTML elements styled and enhanced with extensible classes, and an advanced grid system. This is covered in detail in the section Bootstrap with CSS Components: Bootstrap contains over a dozen reusable components built to provide iconography, dropdowns, navigation, alerts, pop-overs, and much more. This is covered in detail in the section. Layout Component JavaScript Plugins, Bootstrap contains over a dozen custom jQuery

plugins. You can easily include them all, or one by one. This is covered in

details in the section Bootstrap Plugins Customize, You can customize Bootstrap's components, LESS variables, and jQuery plugins to get your very own version.

**CSS**

CSS is an abbreviation for Cascading Style Sheets. CSS works with HTML and other Markup

Languages (such as XHTML and XML) to control the way the content is presented. Cascading Style Sheets is a means to separate the appearance of a webpage from the content of a webpage. CSS is a recommendation of the World Wide Web Consortium(the W3C). The W3C is a consortium of web stakeholders: universities, companies such as Microsoft, Netscape

and Macromedia, and experts in many web related fields. The presentation is specified by styles, which are presented in a style sheet. If you're familiar with word processing programs

like Microsoft Word, you have probably played around at least a little bit with styles. For example, when you want to make the headline text of your document big and bold, the hard way to do it would be to select the text, select a font face and weight, and select the color. The easier way to do it (presuming your document has more than one headline) is to create a "rule", or style, for all the headlines in your document. Then all you have to do is to make one rule, and keep on applying that to all your headers.

CSS in its most basic form works exactly like this. Instead of using <FONT> tags over and over again to control little sections of your page, you can establish some rules to apply globally, to a single page or all the pages on your site. CSS is a great time saver.

**JQuery**

It might seem a bit silly to speak about the merits of jQuery within this cookbook,

especially since you’re reading this cookbook and are likely already aware of the merits.

So, while I might be preaching to the choir here, we’re going to take a quick look at

why a developer might choose to use jQuery. My point in doing this is to foster your

basic knowledge of jQuery by first explaining the “why” before we look at the “how.”

In building a case for jQuery, I’m not going to compare jQuery to its competitors in

order to elevate jQuery’s significance. That’s because I just don’t believe that there

really is a direct competitor. Also, I believe the only library available today that meets

the needs of both designer types and programmer types is jQuery. In this context,

jQuery is in a class of its own. Of the notorious JavaScript libraries and frameworks in the wild, I truly believe each has its own niche and value. A broad comparison is silly, but it’s nevertheless attempted all the time. Heck, I am even guilty of it myself. However, after much thought on the topic, I truly believe that all JavaScript libraries are good at something. They all have value. What makes one more valuable than the other depends more upon who is using it and how it’s being used than what it actually does. Besides, it has been my observation that micro differences across JavaScript libraries are often trivial in consideration of the broader goals of JavaScript development. So, without further philosophical ramblings.

Statement of Purpose

The main objective of objective of our system is to give ease the students to record their information, edit and/or update and to view easily the prerequisite subjects they have taken/dropped.

The design of our website is minimal for the students can easily viewing their account/information and for the administrator can easily manage, categorize, gather information from the students.

The development of this system is based on how the students interact with the GUI of the system

Significance of the Study

As a developer and researcher of this system it will help us how to solve the problem related in the student’s information; it will enhance the skill in developing a system; become professional in building web pages moreover in coding; it will enhance the artistic in designing a website that are capable in student’s need.

The students will serve as a beneficiary of this system because they will have a system that can easily record their information through online. This system is available online for hassle-free service for student.

The school will also benefited in this study because it will help them to organize the students information that they need and they can easily manage every student’s profile/prerequisite subject.

Definition of terms

For a better understanding of this study, the following terms are operationally defined.

**Record**. all documented information, regardless of its characteristics, media, physical form, and the manner it is recorded or stored. Records include accounts, agreements, books, drawings, letters, magnetic/optical disks, memos, micrographics, etc. Generally speaking, records function as evidence of activities, whereas documents function as evidence of intentions.

**Online**. connected by computer to one or more other computers or networks, as through a commercial electronic information service or the Internet.

**System**. an organized, purposeful structure that consists of interrelated and interdependent elements (components, entities, factors, members, parts etc.). These elements continually influence one another (directly or indirectly) to maintain their activity and the existence of the system, in order to achieve the goal of the system.

**Information**. knowledge obtained from investigation, study, or instruction

**RFID**. stands for Radio-Frequency IDentification. The acronym refers to small electronic devices that consist of a small chip and an antenna. The chip typically is capable of carrying 2,000 bytes of data or less.

**Database**. A database is a collection of information that is organized so that it can easily be accessed, managed, and updated. In one view, databases can be classified according to types of content: bibliographic, full-text, numeric, and images.

**Prototype**. In software development, a prototype is a rudimentary working model of a product or information system, usually built for demonstration purposes or as part of the development process. In the systems development life cycle (SDLC) Prototyping Model, a basic version of the system is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed.

**Server**. A server is a computer program that provides services to other computer programs (and their users) in the same or other computers. The computer that a server program runs in is also frequently referred to as a server.

**Developers**. a person or thing that develops or innovates.

**References**

<http://www.businessdictionary.com/definition/record.html>

<http://www.dictionary.com/browse/online>

<http://www.businessdictionary.com/definition/system.html>

<https://www.merriam-webster.com/dictionary/information>

[*http://www.technovelgy.com/ct/technology-article.asp*](http://www.technovelgy.com/ct/technology-article.asp)

[*http://searchsqlserver.techtarget.com/definition/database*](http://searchsqlserver.techtarget.com/definition/database)

[*http://searchmanufacturingerp.techtarget.com/definition/prototype*](http://searchmanufacturingerp.techtarget.com/definition/prototype)

[*http://whatis.techtarget.com/definition/server*](http://whatis.techtarget.com/definition/server)

[*http://www.dictionary.com/browse/developer*](http://www.dictionary.com/browse/developer)

Scope and Limitations

This system built to determine the status of the subject that have taken by the student; to gather/record information of the students; to organize the records of student in a department. The system will help the students to input their information easily through online. The students, professor, organization, and department of Olivarez College especially in the College of Computer Studies.