Object Oriented Analysis & Design

Week 2

Introduction to Object Oriented Analysis and Design

Lecturer: Dr. Msagha J Mbogholi, PhD

Flashback from Lesson 1

- The object oriented paradigm can be traced as follows: procedural paradigm → modular (structured) paradigm → data abstraction paradigm → object oriented paradigm
- The foundations (also referred to as the pillars) of object orientation are abstraction, encapsulation, modularity, and hierarchy.
- Object oriented concepts introduced in this chapter were object, class, attribute, operation, polymorphism, component, package and subsystem.
- The strengths of object orientation are based on the foundations; reuse, flexibility and scalability are also key strengths.
- Weaknesses of object orientation are based on the supposed inefficiency and complexity of the process, especially for those schooled in other methods.
- There are several applications of object orientation in fields as diverse as science and medicine, to office automation tools.

Content

- Introduction
- Tradition Approach to System Development
- System Development Methodologies
- System Analysis and Design
- Introduction to the UML

Part 1

Introduction

Introduction

- The solution to most of today's problems is found in software applications and innovations. There are applications ranging from social media to complex scientific ones.
- For example there are applications such as Facebook and Instagram which provide social platforms; software as a service (SaaS) which can be found on platforms provided by Google and Microsoft, and many more.
- On the scientific front there are applications that help to track diseases and also in the field of robotics to provide both civilian and military solutions. There is also machine learning (ML) which is emerging as a field with a lot to offer.
- In business there are many solutions that help to automate processes and make the overall business system more efficient. This has led to the emergence of several enterprise resource planning (ERP) systems such as SAP and Oracle among others. The demand has pushed all these vendors to move their products to the cloud in order to expand their reach and offer customized solutions.
- But the question that begs an answer is, how do these vendors and other software developers know there is a need for a certain product or service in the business or science environment?

Introduction (cont'd)

- In most cases these vendors study business processes and how they can be made more efficient. For example the supply chain has common elements for most organizations, with minimal customization required for specific domains.
- However, some organizations prefer to hire developers who will develop bespoke systems for them; indeed this has been the case over the years prior to the development of applications like SAP and Sage which now allow for customization.
- The problem in the past was that buying COTS (commercial off the shelf) products resulted in the organization having to adjust internal processes to the software, which was an expensive and often frustrating process. What was needed was development to accommodate and streamline processes, and not the other way round.
- Enter software engineering. IEEE defines software engineering as:
- "(1) The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software.
- (2) The study of approaches as in the above statement." (as cited in tutorialspoint.com)

Introduction (cont'd)

- Essentially this description states that software engineering is the application of engineering to software. In engineering there is a systematic, disciplined, quantifiable approach used to develop different solutions.
- For example a civil engineer trained in Europe or Asia understands the way to design and build roads in the same manner; the only difference will be in the customizations required in each environment. The same applies to automotive engineers and even electrical and energy engineers.
- In software we have an interest in following an approach that is systematic, disciplined and quantifiable in the development of solutions. There are several approaches that have emerged and evolved over time.
- In this lesson these approaches are discussed. The learner may wonder what all this has to do with the subject of object oriented analysis and design right?

Introduction (cont'd)

- Well, the lesson describes the approaches and how they are related to analysis and design in general. There is more than one approach to design as shall be seen. After describing the general software development lifecycle (SDLC),implementations of it are described using three main models. These models are used practically in methodologies such as Rapid Application Development (RAD), Agile, Prototyping, Extreme Programming (XP) and Scrum.
- The lesson then turns its attention to the object oriented approach and what makes it different from the traditional approach; further why it is the paradigm of choice in today's software engineering environment.
- Finally an introduction is made to the Unified Modelling Language (UML).

Part 2

Traditional Approach to System Development

2.0 Introduction

- When studying information resource management in an organization one of the key challenges is where the information system (IS) will fit in the greater organization goals.
- Organizations go into business in order to make a profit and management sees IS as an expense rather than a benefit. The CxO (in this case either CIO or CTO) in charge needs to justify the cost of all the equipment and the accompanying software (which isn't easy especially when there's an accountant sitting in that meeting).
- The bone of contention is how will the system contribute to the organization's goals and strategy? The only way the CxO can show this is to show where costs will be reduced and operations or processes streamlined courtesy of the suggested software. A cost benefit analysis (CBA) is in order in this case.
- The other challenge is that some of the benefits are intangible rather than tangible; for example, ease of use may boost employee morale which in turn leads to higher productivity.

Introduction (cont'd)

- The company hired to develop the software must ensure that the software indeed does improve the company processes, and also adds value. Further the software has to be delivered within a budget and in a timely manner. The identification of what the software will do is also a challenge since the users (and management) aren't tech savvy; this presents a unique challenge for the systems analysts who will listen to the organization's needs and transform them into the eventual system that will fulfill those needs.
- Software engineering provides a general approach that is used for the process, from the identification of the need to the eventual commissioning of the system. This approach is called the software development life cycle (SDLC).

2.1 Software Development Lifecycle (SDLC)

- Imagine that you have bought an 8 acre piece of land by the beach. You wish to build a nice resort where people can come and enjoy sunsets and sunrises by the beach, swim, and have accompanying candle lit dinners by the ocean. Sounds sublime doesn't it?
- But where do you start? You have to engage an architect to bring out the idea you have…is it a one storey building, or more? How big? How wide? Then you have to bring in a QS (quantity surveyor) to tell you how much the materials will cost, an interior designer, an electrical engineer, a water engineer, ad infinitum.
- It is a process that will you will follow in order to actualize your vision of the resort. This is exactly what software development entails in a nutshell.
- You start with a vision (what you want the software to do) and engage a professional who will help to actualize that vision by going through a series of steps (just like what you have done with the resort plan).
- The SDLC was developed to provides a series of steps (a framework if you like) that guides the development of software from concept to actualization. In some literature you may find it being referred to as systems development lifecycle but it is the same lifecycle nonetheless.

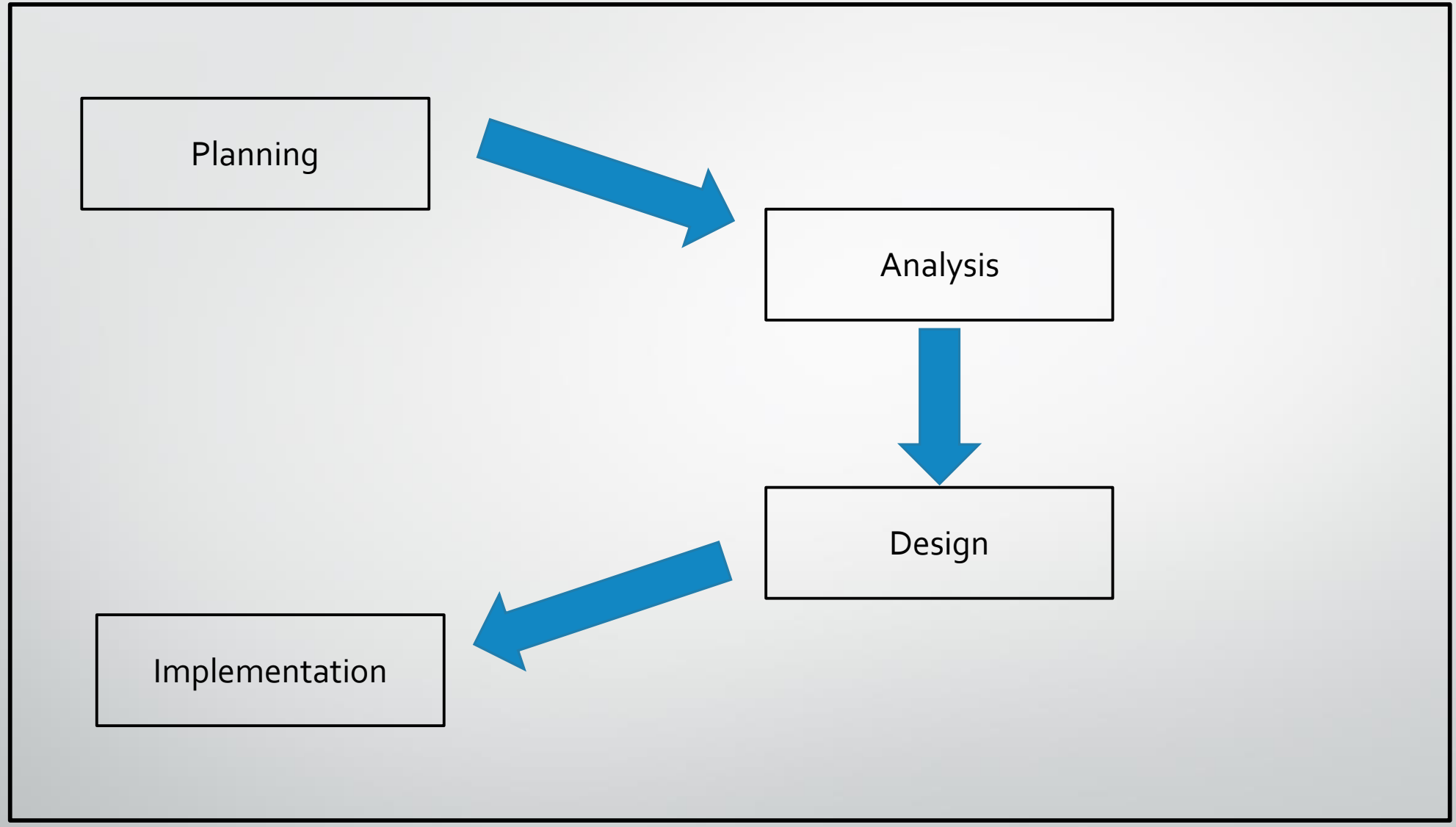

Fig 1. Systems Development Life cycle

3.1 SDLC (Cont'd)

- The SDLC is presented differently in separate literature (see for example, https://www.clouddefense.ai/blog/system-development-life-cycle, https://www.tutorialspoint.com/sdlc/sdlc_overview.htm, and https://www.javatpoint.com/software-engineering-software-development-life-cycle) ; however, all presentations stem from four simple steps.
- The differences come about due to expounding on the steps to create more steps in the different adaptations. Fig 1 describes the four steps in the SDLC.
- The original form of the SDLC was described by Daniels and Yates (1971). This is shown in fig 2. In their description they stated what needed to be done in each phase; a comparison of both figures (1 and 2) shows the close resemblance and the fact that even the original explanation is captured succinctly in fig 1.
- As describe earlier each of the phases involves a series of steps within it; this is what has led to different presentations of the SDLC in literature. However, most important is that the steps in the cycle are taken sequentially and conform to the IEEE definition of software engineering. The completion of the cycle indicates the delivery of the required product.

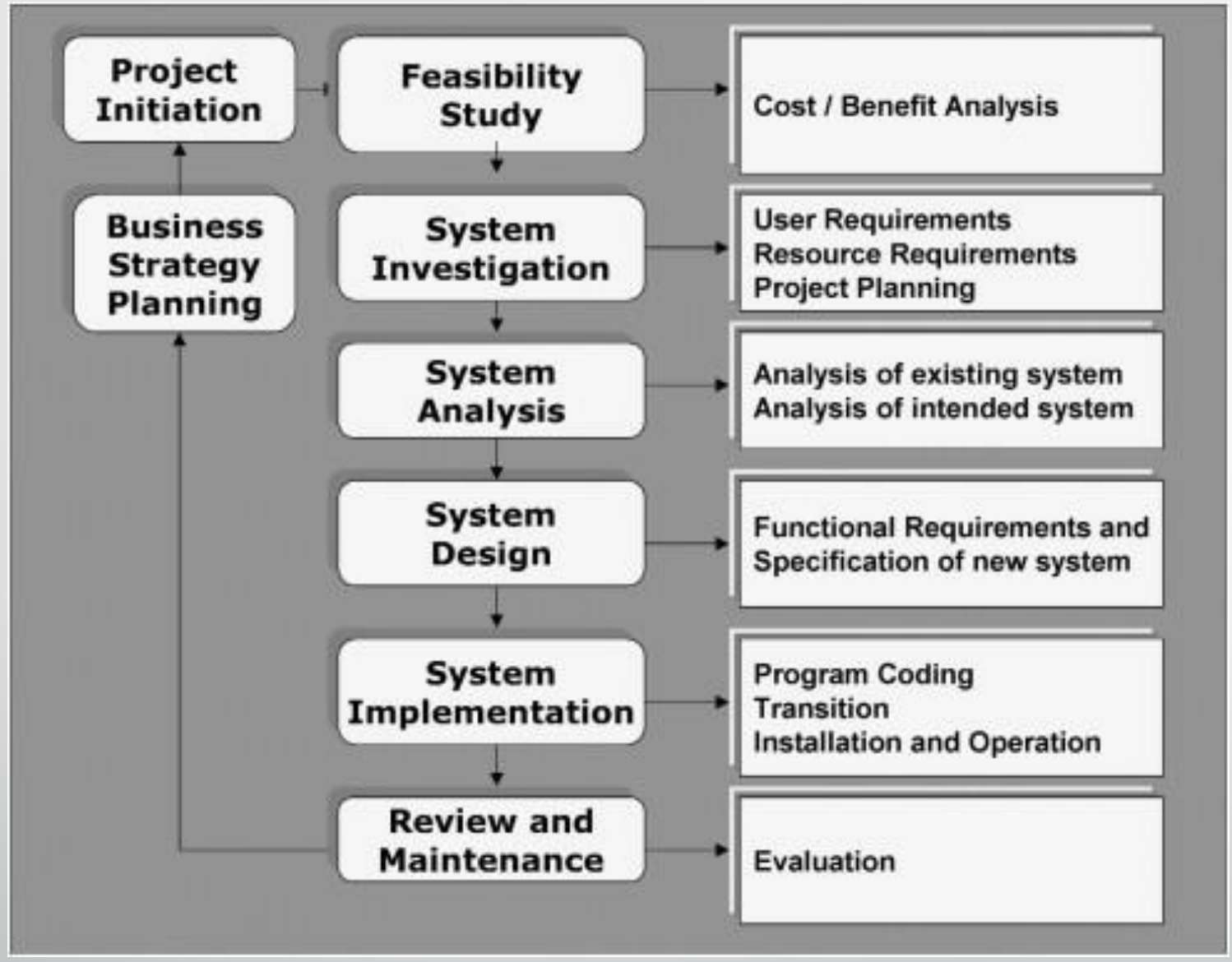

Fig 2. conventional SDLC model (Stefanou, 2003. Adapted from Daniels and Yaetes, 1971)

3.1.1 Planning

- An organization has two ways in which to acquire a system: buy COTS or develop bespoke (either through an in-house department or hire another company specializing in development to do it). Either way it is best undertaken as a project so that it can be monitored and controlled.
- This means that in most instances the need for the software has to be identified first. This can come from a user department or from the management. This generates a sequence of steps as follows:
- A department feels there is a need for software to improve or automate a process; they make a request for it to the relevant authority (management or IS department).
- The authority meets to discuss the (de)merits of the proposed system: does it add value, improve processes, increase revenue?
- If the answer is yes, then a feasibility study is done which should also involve a CBA (cost benefit analysis). If it is found to be cost effective then the request is approved.

3.1.1 Planning (cont'd)

- After approval the project team is notified (in-house or outsourced) and they can now swing into action.
- The project manager now develops a project plan describing how it will be implemented and the different steps of the SDLC to be undertaken.
- Based on requirements this is where the project manager selects the different team members who will take part in the project.
- The project plan is the document that will describe how the team will go about the project (description, timelines, budget, and so on).
- Once this is done the project can move to the next phase.

3.1.2 Analysis

- The Merriam-Webster dictionary defines analysis as "a detailed examination of anything complex in order to understand its nature or to determine its essential features". This is what is involved in this phase.
- Let us bear in mind that the proposed system is supposed to improve some processes or automate them. That means that there is already an existing system doing this, be it manual or also automated.
- This phase can be summarized as "who, what, where and when" (of the proposed system). That is to say:
- Who – who is going to use the proposed system?
- What – what is the proposed system supposed to do?
- Where – where is the proposed system supposed to be used (department, function, or enterprise wide)?
- When – when is the proposed system supposed to be used (end month, for example payroll, when billing is being done, during manufacturing, throughout the day, on weekends, and so on)?

3.1.2. Analysis (cont'd)

- Essentially there are three steps in this phase that help to identify the opportunities in order to come up with a concept for the proposed system. These steps are (Dennis, Wixon and Tegarden, 2015):
- Analysis strategy – what is the problem with the current way (system) and consequently, how do we design the new system?
- Requirements gathering – use of appropriate instruments (for example interviews) to determine requirements (expectations) of the users. This will help in coming up with a system concept. The system concept helps to develop the business analysis models that will "map" the business processes involved to the proposed system. This will enable the team to see how the system will function if implemented in the business environment.
- System proposal – the system concepts, models, and analysis make up this document.
- The proposal is then presented to the relevant authorities for approval before moving to the next phase.

3.1.3 Design

- Recall that in the analysis phase we asked the main question of "what"? Meaning what is the proposed system supposed to do? This information was captured in the system proposal document.
- The design phase is about "how". That is to say, how is the system going to be implemented in terms of hardware, software, standalone or network enabled), what type of system (files, databases and so on). The result of this phase is a document called the system specification (and rightly so named), which is handed over to the programming team for implementation. It will contain the following details (Denis, Wixon and Tegarden, 2015):
- Design strategy – COTS, in-house or outsource the system?
- Architecture design – hardware, software, network infrastructure (if applicable), interfaces, reports, and input forms.
- Database specifications – data details, that is, what information needs to be stored and where
- Program specifications – what program (s) need to be written and their purpose (s).

3.1.4 Implementation

- As the name implies this is the phase where the system is actually built by the programming team, and eventually commissioned. The phase involves:
- Software construction – the programmers write the code that will implement the system according to the specifications in the previous phase. The code is written in a specific coding language and interfaces and tables (database) integrated with the forms.
- Testing – the program is then tested according to software engineering practice (module and integration testing) to see that it is functioning as intended. At this stage debugging is also done. Testing also involves users and can take more time than the construction sub phase.
- Installation – after thorough and satisfactory testing the system is ready to be commissioned into the live environment. What remains is to have a training plan for the users and to manage change (human beings are resistant to change by nature, while other users might resist for other reasons).
- Maintenance – this phase is normally taken as a phase in itself in most literature. It costs the company up to 20% of the total purchase price of the software annually. The reason for this is that maintenance accommodates changes that will need to be done to the system over time (due to changing business needs) and also for updating and debugging.

Part 3

System Development Methodologies

Introduction

- A method is a formal way of doing something. In mathematics exams for example, marks are awarded for using the right method (approach) even if the answer is wrong. The word methodology is derived from method, and refers to a formal way of doing something.
- Software development methodologies (SDM), consequently are different formal ways by which software is developed using the SDLC as an implementation model (guide or backbone). This is to say that SDM are different ways of implementing the SDLC.
- SDM are categorized in different ways depending on the focus of the one categorizing them. It is better to classify the approaches that implement the different SDM models as we have done in this section, rather than trying categorize the SDM models themselves (this will become clearer as we progress).
- However, from a system analysis and design perspective SDM can be classified based on the type of focus they support. There are 3 broad types of focus:
    - Process
    - Data
    - Object

Introduction (cont'd)

- When analyzing and developing a system you may choose to focus on the processes of the system and develop on that; this is called process centered design. The focus is on the processes first then figure out the data later.
- In data centered approach a data model is developed. In this case focus is on the data itself and then figure out the processes that will support this later.
- The object oriented approach is a hybrid of the process and data approach; both are combined into one object model (which is why it is so powerful).
- In this section four methodologies (models) are described and their applications.

3.1 Waterfall Methodology

- This was the first methodology developed from the SDLC.
- In fact quite a few literature use the name SDLC to mean the waterfall methodology and vice versa! The reason for this is that the steps in this model are a replica of those in the SDLC.
- The steps of the waterfall model are shown in fig 3. They are:
- Requirements, analysis, design, specification, implementation, testing, deployment, maintenance.
- The phases are same as SDLC with the difference being on the splitting of some SDLC sub phases into main phases. For example implementation, testing and deployment all fall under implementation in the SDLC, while design and specification are also in the design phase of the SDLC.
- One worthwhile split in the waterfall model is the naming of maintenance as a distinct phase. As described in the discussion on the SDLC maintenance is crucial to the ongoing health of the system over time. Without maintenance the system can lose its purpose in a short period.
- Table 1 presents the advantages and disadvantages of the waterfall methodology.

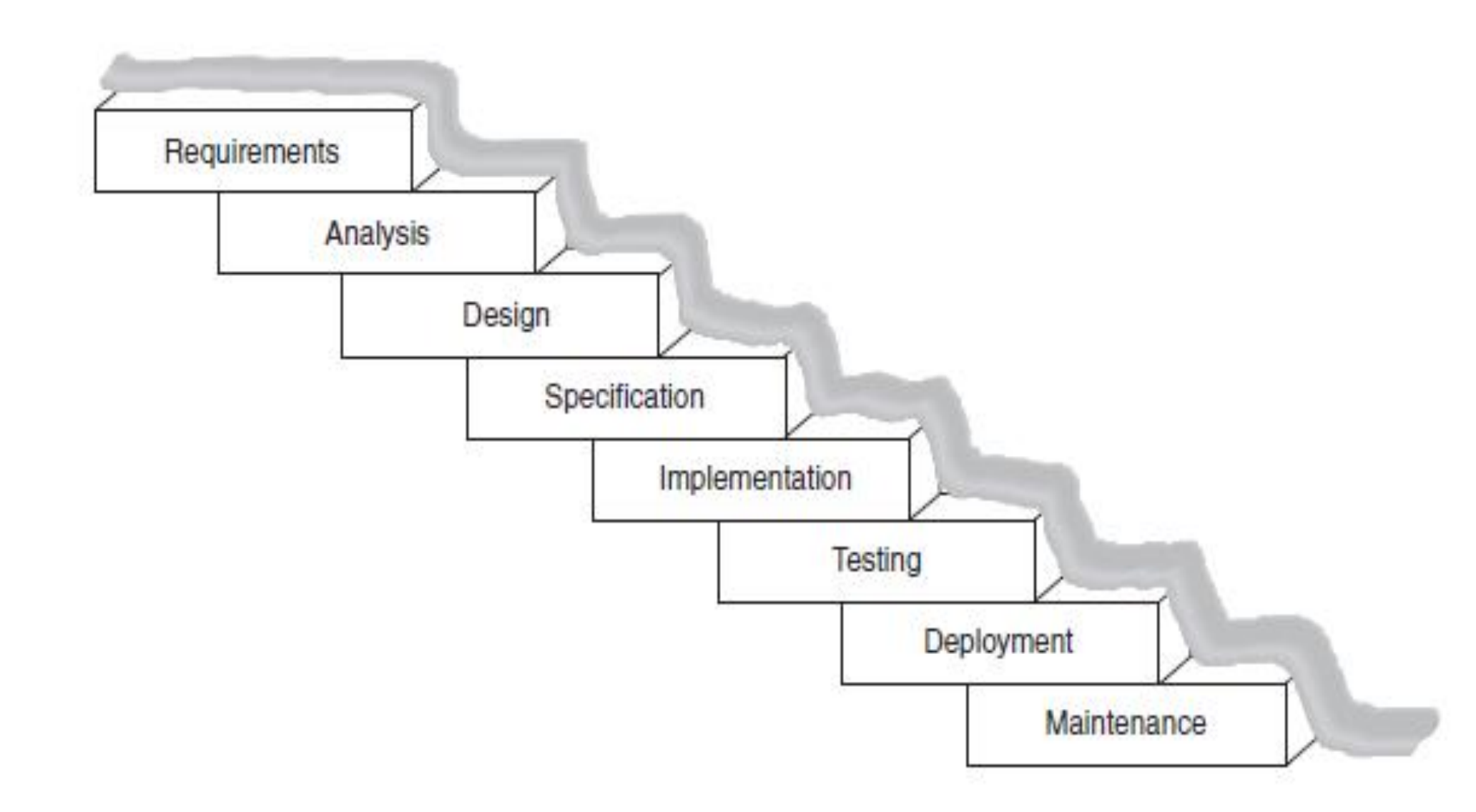

Fig 3. Waterfall methodology (O'Docherty, 2005)

Advantages	Disadvantages
This model is easy and simple and to understand and utilise.	It is extremely difficult to proceed back and correct something that was not well-thought-out in the concept stage once an application is in the testing stage.
It is quite a simple to manage due to the rigidity of the model – each phase has a specific review process and deliverables.	No effective software is produced until late throughout the life cycle.
In this model, phases are prepared and completed one at a time. Phases do not overlay.	Not an efficient model for object-oriented and complex projects.
The waterfall model operates well for smaller projects where requirements are precisely defined and very well understood.	High amounts of uncertainty and risk.

Table 1. Advantages and disadvantages of waterfall methodology (Prasana, 2022)

3.2 Spiral Methodology

- This development methodology imitates a spiral. The steps, however, are the same as for the waterfall methodology; the implementation is not.
- This methodology begins with an initial capture of the requirements; then an analysis is done based on the captured requirements; this is followed by a system design that captures these requirements; thereafter some code is written based on the system design and tested (informally). This is then presented to the users for comments.
- Of course this first round did not capture all the requirements; but it was crucial since the development team now has a better understanding of what the system should do.
- The spiral is then repeated a second time repeating the process as more requirements are captured and more details are captured in the whole process.
- By the third or fourth spiral all the requirements will probably have been captured and the design and implementation is completed. This allows for the system to undergo a comprehensive testing phase before being deployed into the live environment.
- Fig 4 demonstrates the process (showing where the spiral begins, marked with an arrow).
- Table 2 describes the advantages and disadvantages of the spiral development methodology.

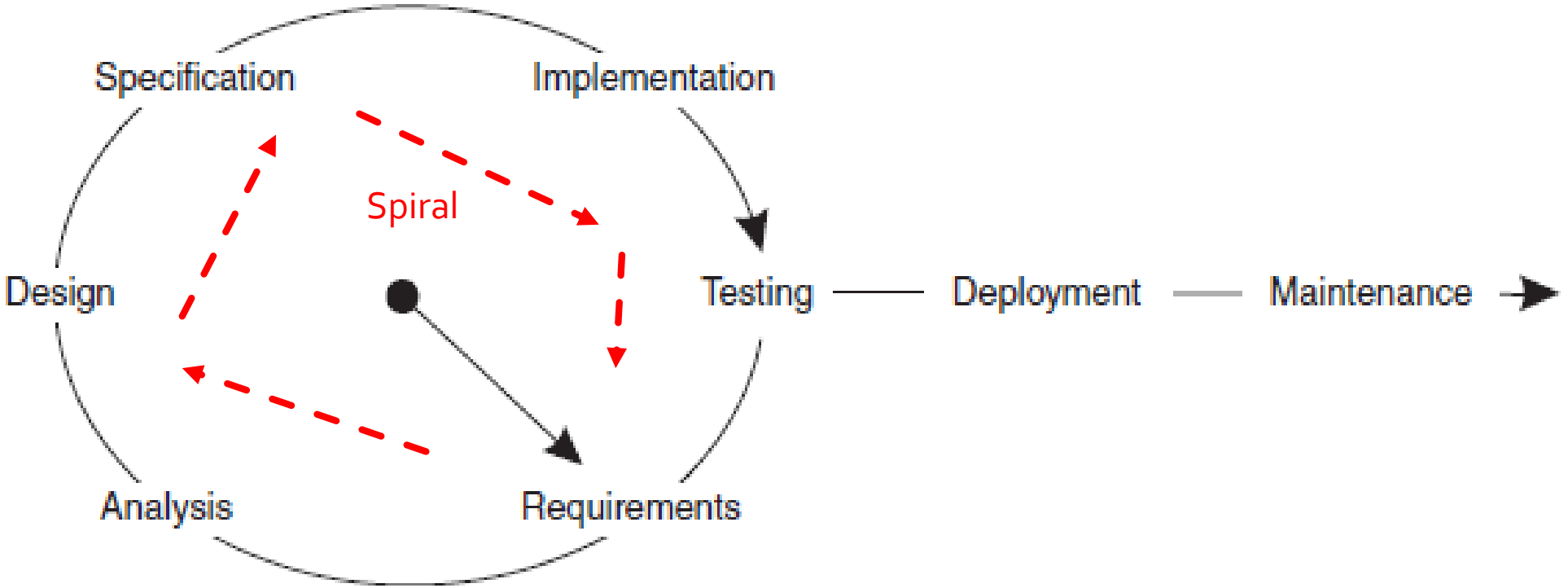

Fig 4. Spiral Development Methodology (Adapted from O'Docherty, 2005)

Advantages	Disadvantages
The risk management feature is one of the best development characteristics of the spiral model.	The spiral model is difficult to use since the volume of documentation required in its initial stage is vast.
The spiral model is suitable for intricate and extensive projects.	Developing a spiral model is very expensive and small projects may find it difficult to use.
Replacements and other demands for project requirement in the spiral model are docile.	The spiral model depends on risk analysis and without a highly experienced team developing this project might be impossible.
Clients can observe the full process of development from the initial stage to the completion stage hence receiving satisfaction.	Management of the spiral model is quite difficult due to the risk factors that are unknown to developers at the starting stage.
The early estimation cost of the project can be computed in the spiral model phases.	The spiral model is not user-friendly especially for projects with an unambiguous SRS.

Table 2. Advantages and disadvantages of spiral development methodology (Prasana, 2022)

3.3 Iterative Methodology

- As may have been inferred from the spiral method, it is quite time consuming and expensive to go round the whole spiral every time in order to gain comprehensive understanding of the requirements.
- The iterative methodology finds a way around this. This methodology allows you to go back to a phase without having to go around the whole spiral. It means that it is possible to iterate between phases in order to improve the final system.
- You can look at iteration like a while loop in programming (or like you repeating something until it fulfills a condition like say, being perfect).
- What this means is that you can apply this methodology to any other methodology, for example the waterfall methodology.
- Fig 5 shows the application of iterative methodology in a spiral, while table 3 shows the advantages and disadvantages of iterative methodology.

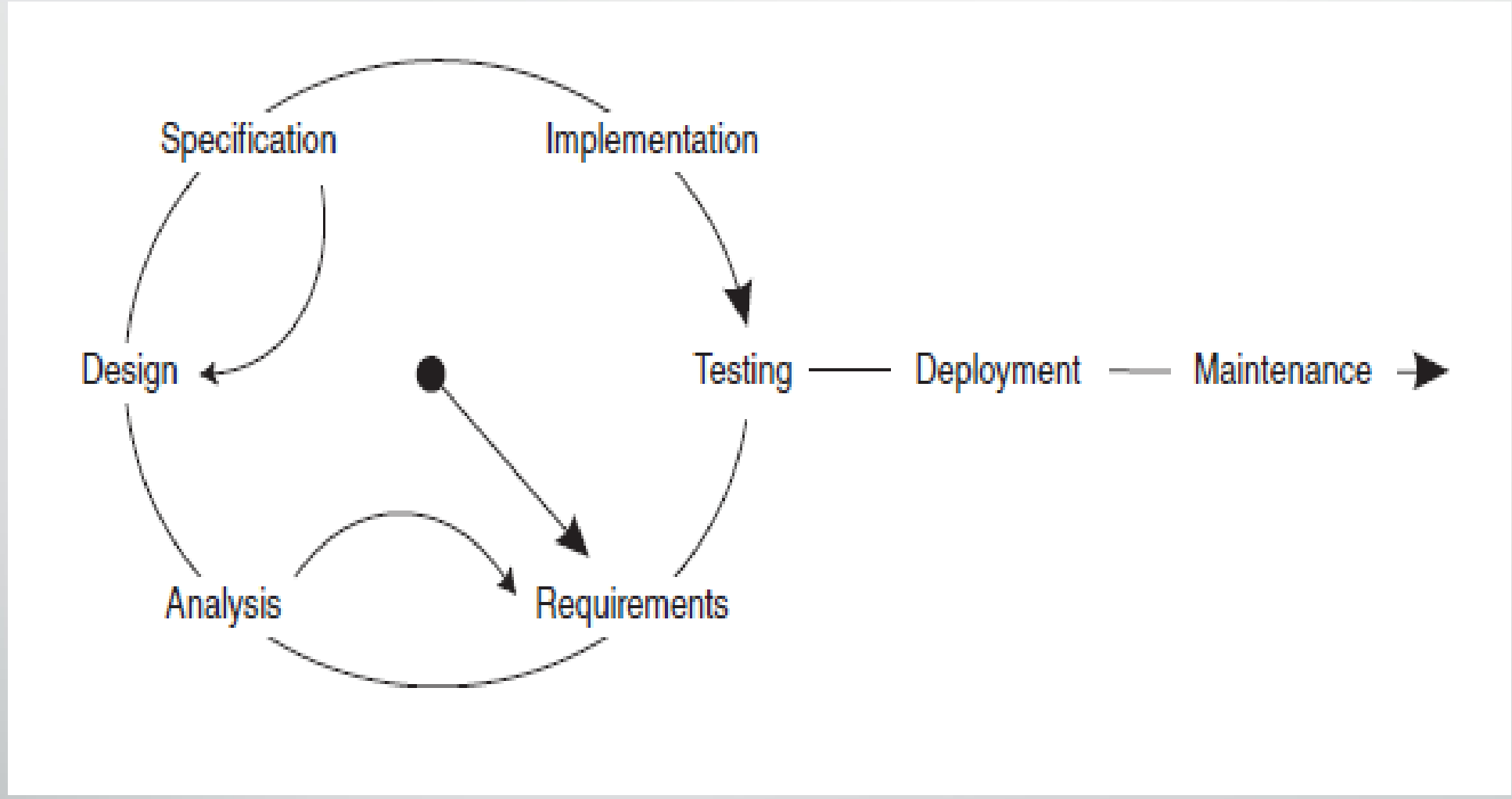

Fig 5. Iterative methodology (in a spiral) (O'Docherty, 2005)

ADVANTAGES	DISADVANTAGES
Some functions can be quickly developed at the beginning of the development lifecycle	Iterative model requires more resources than the waterfall model
The paralleled development can be applied	Constant management is required
The progress is easy measurable	Issues with architecture or design may occur because not all the requirements are foreseen during the short planning stage
The shorter iteration is – the easier testing and debugging stages are	Bad choice for the small projects
It is easier to control the risks as high-risk tasks are completed first	The process is difficult to manage
Problems and risks defined within one iteration can be prevented in the next sprints	The risks may not be completely determined even at the final stage of the project
Flexibility and readiness to the changes in the requirements	Risks analysis requires involvement of the highly-qualified specialists

Table 3. Advantages and disadvantages of iterative methodology (Osetsyki, 2022)

3.4 Incremental Methodology

- Have you ever bought software with versions? Today you buy version 2.0; in a couple of months the vendor alerts you that there is now version 2.1 with some additional features; 3 months later there's version 2.2, and so on.
- This is the application of incremental methodology. In this case the users are given an initial version to work with which has all the critical functionality required. Then newer versions are produced successfully with more functionality as per the requirements.
- Therefore we may have the first version capturing critical functionality, with successive versions adding functionality, resulting in a final version that captures all the functionality.
- Fig 6 demonstrates incremental methodology; the emphasis is that once a version has been created there is no going back to it. All subsequent versions will undergo all the phases and add more functionality to the previous version. Table 4 describes the advantages and disadvantages of incremental methodology.

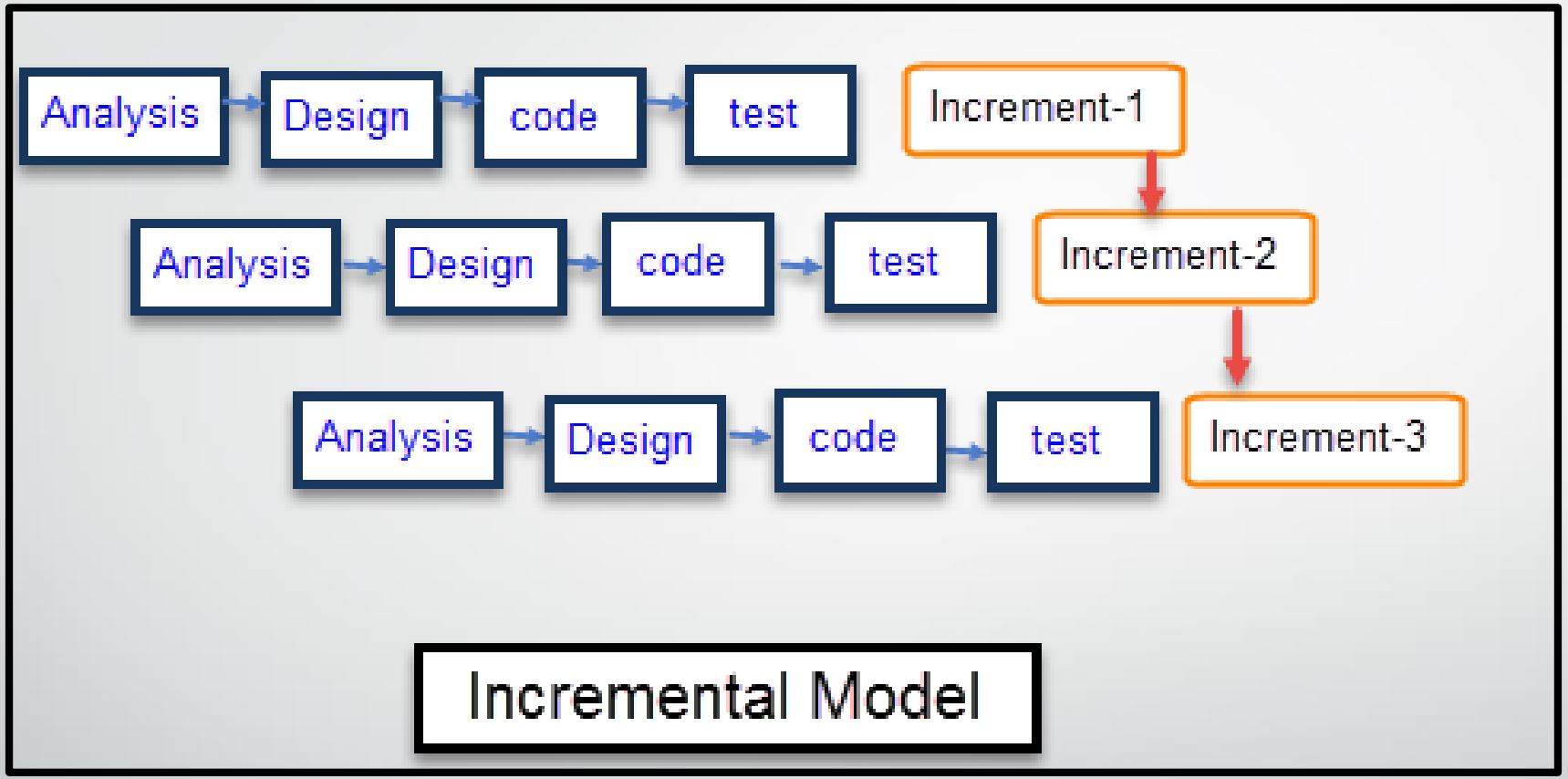

Fig 6. Incremental methodology (Martin, 2022)

Advantages	Disadvantages
•The software will be generated quickly during the software life cycle	•It requires a good planning designing
•It is flexible and less expensive to change requirements and scope	•Problems might cause due to system architecture as such not all requirements collected up front for the entire software lifecycle
•Throughout the development stages changes can be done	•Each iteration phase is rigid and does not overlap each other
•This model is less costly compared to others	•Rectifying a problem in one unit requires correction in all the units and consumes a lot of time
•A customer can respond to each building	
•Errors are easy to be identified	

Table 4. Advantages and disadvantages of incremental methodology (guru99.com)

3.5 Applications of SDM

- The methodologies discussed so far each have their own unique advantages (and disadvantages). However, it is possible to combine the models to produce something arguably more effective than when they are implemented individually. In this section some of these applications are discussed briefly. For more detailed explanations the software engineering course will suffice:
- Prototyping – in prototyping a product (called a prototype) is developed capturing the key functionalities of the proposed system. It is then tested by the customers who provide feedback to improve on it till the final product is accepted. It is possible to build an incremental prototype though (further reading on this is encouraged).
- Rapid application development (RAD) – it is based on both prototyping and iterative model. The idea is to produce a prototype and then improve on it iteratively like a spiral.
- Agile development – this is based on both the iterative and incremental approach. It emphasizes simple iterative application development. Scrum – it also is based on both iterative and incremental approach. It is actually an implementation of agile methodology.
- Extreme programming (XP) – It is an agile development framework.
- Lastly table 5 offers a comparison of the three main models of software development.

Model	advantages	disadvantages
Waterfall Model	1. Easy to understand and implement 2. Reinforces good habits: define-before-design and design-before-code. 3. Identifies deliverables and milestones 4. Works well on mature deliverables	1. Real projects rarely follow sequential approach 2. Uncertainty at the beginning of the development 3. No working version of the system until very late
Incremental Model	1. Divides project into smaller parts 2. Creates working model early 3. Feedback from one phase provides information for the next phase 4. Very useful when more staffing is unavailable	1. Users need to be actively involved in the project. 2. Communication and coordination skills are central process 3. Informal requests for improvement for each phase may lead to confusion 4. It may lead to scope creep
Spiral Model	1. Designed to include the best features form Waterfall and Prototyping Model 2. Good for large and mission-critical projects 3. Introduces risk assessment as a new component	1. Can be a costly model to use 2. Risk analysis requires specific expertise 3. Project's success is highly dependent on risk analysis phase 4. Doesn't work well for smaller projects

Table 5. Comparison of software development methodologies (Subair, 2014)

Part 4

System Analysis and Design

Introduction

- In the previous sections of this lesson we have discussed the different methodologies (or processes as they are called in some literature) that are used to develop process.
- It should be apparent that even though all the steps are crucial, the devil in the details is in the analysis and design stages. This is because this is where an understanding of the requirements is essential and conversion to the system design happens. The role of the programmer is to just write the design in code; so if the design is wrong then it follows everything else will be wrong.
- In the software development world there are two main approaches to analysis and design; these are structured and object oriented. In this section an introduction to the structured method is discussed before examining the object oriented approach.
- It is important to understand the difference between these two approaches. It is by understanding the difference that the learner can appreciate the strength of the object oriented approach vis a vis the structured approach.
- The elements of object orientation are also discussed together with the advantages of object oriented analysis and design.

4.1 Structured system analysis and design (SSADM)

- This approach uses the waterfall model. It is designed to emphasize user involvement and so the users have to sign off at each stage of the process. Due to the intense user involvement this process produces simple diagrams that they (users) can understand at each stage of the process.
- SSADM focuses on processes and data in coming up with the eventual design. There are 3 models that are focused on (techopedia.com):
- Logical data model – this involves classification of the data via the entity relationship diagram (ERD). The ERD shows all the entities in the proposed system and how they are related to each other.
- Data flow model – this involves the use of the dataflow diagram (DFD). The DFDs show the process flow and also how and where data will be stored.
- Entity behavior model – shows how the different entities behave in the proposed system.

4.1 SSADM (cont'd)

- Focus on analysis and design reveals the following:
- Analysis – the main diagrams used in this phase are the DFD, ERD and the data dictionary. The latter describes (defines) the data elements that are not found in the DFD.
- Design – the two main instruments in this phase are the structure chart and pseudocode. The structure chart is a top down design of the system broken into components. It shows the relationship between components (modules) and what data needs to be passed between them. Pseudocode is the actual implementation of the proposed system written in a non specific programming language; that is to say, it is written in English, but in short form. There is no standard pseudocode "language".
- Other diagrams that are used to support the above are decision trees and decision tables.
- A key advantage of SSADM is the user involvement; since they are involved in every phase of the waterfall model there is slim possibility of having to go back to any step. However, because of this it is a very time consuming process and normally takes a long time from the requirements to the actual development of the system. Who wants that in today's fast moving technological world?

4.2 Object oriented analysis and design (OOAD)

- Enter object oriented analysis and design (OOAD). This is the approach of choice in software engineering today. Indeed most languages (covered in another lesson) are based on the concept of objects. Let us examine a few definitions:
- Object-oriented analysis and design is using an object-oriented approach to building conceptual and logical models of the system. (Ashrafi, N and Ashrafi, H, 2014)
- Object Oriented Analysis (OOA) is concerned with developing requirements and specifications expressed as an object model (population of interacting objects) of a system, as opposed to the traditional data or functional views. (Software Engineering Institute)
- The object oriented approach is built on the foundations of object orientation described in lesson 1; abstraction, encapsulation, modularity, and hierarchy. These are all based on the object oriented paradigm, which is based on real world entities called objects.
- The major strength of OOAD is that the analysis and design is based on representation of the actual objects to be used in the system; that is to say, a model of the actual objects is used to develop the system.

4.2 OOAD (cont'd)

- OOAD is based on both the incremental and iterative models.
- Why are models used in OOAD? It is because models simplify reality, making it easier to understand (grasp) concepts. Models are used in literally every sphere of life to increase understanding. For example, an architect will show you a model of the building that will be coming up; when you see it you can visualize and understand it better. When you visit a cardiologist s/he may have a model of the heart on the table; they use this to explain what goes on in the heart (aorta, ventricles and the rest). Similarly models are used in different spheres of engineering to make complex concepts easier to understand.
- Summarily models help to visualize complex concepts, enable templates to be built, allows designers to come up with specifications, and they provide for easier documentation in the long run.

4.2 OOAD (cont'd)

- Startertutorials.com describe 4 principles of modelling applicable to OOAD:
- "The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped.
- Every model may be expressed at different levels of precision. (or abstraction)
- The best models are connected to reality.
- No single model is sufficient. Every non-trivial system is best approached through a small set of nearly independent models."
- OOAD takes the modeling concept to its very core. By using diagrams to represent the objects and showing the relationships between them it makes it that much easier to implement the system. Since each object has attributes and behavior, then they can communicate with each other using messages (as you do in the real world). This way there is no need to focus on processes and data like in SSADM; we only need to focus on what the attributes and methods of each class are, and which messages they can pass to each other.

4.2.1 Elements of OO Methodology

- There are 3 key elements to every OO methodology:
  - Notation – this term is used to refer to symbols that are used to express something else, for example music. In OO the notations used are found in the UML (described in the next section) and they are used to express the model being built.
  - Process – describes the steps that are taken to create the system. As described earlier OO uses both the incremental and iterative processes (methodologies).
  - Tools – these are the software used to support the analysis and design phase. Some good tools to use are StarUML and Rational Rose.

4.3.2 Advantages of OOAD

- There are several advantages associated with the use of OO methodology:
- Reduced maintenance – with the use of objects maintenance becomes easier since it is focused on the objects themselves; further since the code can be reused there is no need to write it over and over again.
- Real – world modelling – the models are based on objects and classes drawn from the real world; it is based on objects rather than data and processes.
- Reliability and flexibility – due to the principles of object orientation, it is easier to add and remove objects, have them inherit from other objects and there is convenient encapsulation.
- Reusability – all the code in the model (system) can be reused in different areas to fulfill objectives of the overall requirements.
- Table 6 offers a comparison of SSADM and OOAD highlighting strengths as well as weaknesses of both methodologies.

Structured Analysis	Object-Oriented Analysis
The main focus is on the process and procedures of the system.	The main focus is on data structure and real-world objects that are important.
It uses System Development Life Cycle (SDLC) methodology for different purposes like planning, analyzing, designing, implementing, and supporting an information system.	It uses Incremental or Iterative methodology to refine and extend our design.
It is suitable for well-defined projects with stable user requirements.	It is suitable for large projects with changing user requirements.
Risk while using this analysis technique is high and reusability is also low.	Risk while using this analysis technique is low and reusability is also high.
Structuring requirements include DFDs (Data Flow Diagram), Structured Analysis, ER (Entity Relationship) diagram, CFD (Control Flow Diagram), Data Dictionary, Decision table/tree, and the State transition diagram.	Requirement engineering includes the Use case model (find Use cases, Flow of events, Activity Diagram), the Object model (find Classes and class relations, Object interaction, Object to ER mapping), Statechart Diagram, and deployment diagram.

Table 6. Comparison of SSADM and OOAD (Hammad, 2022)

Part 5

Introduction to the Unified Modelling Language (UML)

Introduction

- The Unified Modelling Language (UML) is the visual language that is used to express notation in OO methodology. It is not language dependent and is very simple to understand. The UML is also non-proprietary.
- By definition it is "a general-purpose, graphical modeling language in the field of Software Engineering. UML is used to specify, visualize, construct, and document the artifacts (major elements) of the software system." (javatpoint.com)
- The UML is suitable for software intensive systems such as business processes and enterprise wide information systems.
- UML is recommended for use with the processes that are:
- 1) use-case driven
- 2) architecture-centric
- 3) iterative
- 4) incremental
- (Ojo and Estevez, 2005)

5.1 Goals of UML

- The UML provides a general purpose visual modelling language that can be used by all modelers.
- Since it is language independent the UML allows support of specifications and processes that are easily understood by all.
- It can also be used for systems that are non software since it can be used by anyone whether developer or user.
- The building blocks of the UML are:
- Objects – these include all the elements of the systems, depending on the view being constructed
- Relationships – these show how the different objects relate to each other.
- Diagrams – the different diagram groupings that show different views of the model.

5.2 Advantages of UML

- Careerride.com describe the following advantages of using the UML:
- “Provides standard for software development.
- Reducing of costs to develop diagrams of UML using supporting tools.
- Development time is reduced.
- The past faced issues by the developers are no longer exists.
- Has large visual elements to construct and easy to follow.”

Summary

- The Software Development Lifecycle (SDLC) forms the basis for all software development methodologies (processes) (SDM). It has 4 simple steps namely planning, analysis, design and implementation.
- The main SDMs are waterfall, iterative, incremental and spiral. These are applied to Rapid Application Development (RAD), Agile methodologies, Extreme Programming (XP) and Scrum.
- The two approaches to system analysis and design are structured (SSADM) and object oriented (OOAD). The latter offers distinct advantages over the former.
- There are 3 key elements to every OO methodology: notation, process and tools.
- The Unified Modelling Language (UML) is the language used used to specify, visualize, construct, and document the artifacts (major elements) of the software system.

References (1/2)

- Ashrafi, N., & Ashrafi, H. (2014). *Object Oriented Systems Analysis and Design* (1st ed.). Pearson.
- Burleson, D. (n.d.). *Advantages and Disadvantages of Object-Oriented Approach*. Advantages and disadvantages of object-oriented approach. Retrieved October 2, 2022, from http://www.dba-oracle.com/t_object_oriented_approach.htm
- Careerride.com. (2016). *What is UML? What are advantages of using UML?* What is UML? what are advantages of using UML? Retrieved October 2, 2022, from https://www.careerride.com/UML-definition-advantages.aspx
- Dennis, A., Wixom, B. H., Tegarden, D. P., & Seeman, E. (2015). *System analysis & design: An object-oriented approach with Uml*. Wiley.
- Hammad, M. (2022, June 1). *Difference between structured and object-oriented analysis*. GeeksforGeeks. Retrieved October 2, 2022, from https://www.geeksforgeeks.org/difference-between-structured-and-object-oriented-analysis/
- javatpoint.com. (n.d.). *UML tutorial - javatpoint*. www.javatpoint.com. Retrieved October 2, 2022, from https://www.javatpoint.com/uml
- Martin, M. (2022, August 27). *Incremental model in SDLC: Use, advantage & disadvantage*. Guru99. Retrieved October 2, 2022, from https://www.guru99.com/what-is-incremental-model-in-sdlc-advantages-disadvantages.html

References (2/2)

- O'Docherty, M. (2005). *Object-oriented analysis and Design: Understanding system development with Uml 2.0*. Wiley.
- Ojo, A., & Estevez, E. (2005). (rep.). *Object-Oriented Analysis and Design with UML - Training Course* (Vol. 1).
- Osetskyi, V. (2022, July 7). *SDLC models: Agile, waterfall, V-shaped, iterative, Spiral - EXISTEK Blog*. RSS. Retrieved October 2, 2022, from https://existek.com/blog/sdlc-models/
- Pericherla, S. (2018, May 10). *Principles of modeling*. UML Tutorial for Beginners. Retrieved October 2, 2022, from https://www.startertutorials.com/uml/principles-of-modeling.html
- Prasanna. (2022, January 16). *Spiral model advantages and disadvantages: Advantages and disadvantages of using spiral model*. A Plus Topper. Retrieved October 2, 2022, from https://www.aplustopper.com/spiral-model-advantages-and-disadvantages/
- *Software engineering overview*. Tutorials Point. (n.d.). Retrieved October 2, 2022, from https://www.tutorialspoint.com/software_engineering/software_engineering_overview.htm#
- Stefanou, C.J. (2003). Systems Development Lifecycle. *Encyclopaedia of information systems. Elsevier.*
- Subair, Saad. (2014). The Evolution of Software Process Models: From the Waterfall Model to the Unified Modelling Language (UML). 3. 2277-9825.
- Techopedia Inc. (n.d.). *What is structured systems analysis and Design Method (SSADM)? - definition from Techopedia*. Techopedia.com. Retrieved October 2, 2022, from https://www.techopedia.com/definition/3983/structured-systems-analysis-and-design-method-ssadm