Segment 2: Information Gathering

SOFTWARE ENGINEERING PROCESSES

- A software engineering process is the model chosen for managing the creation of software from initial customer inception to the release of the finished product.
- The chosen process usually involves techniques such as:
  - ✓ Analysis,
  - ✓ Design,
  - ✓ Coding,
  - ✓ Testing and
  - ✓ Maintenance
- Different process models use different analysis techniques, other models attempt to implement the solution to a problem in one big-bang approach, while others adopt an iterative approach whereby successively larger and more complete versions of the software are built.

SYSTEMS DEVELOPMENT LIFE CYCLE (SDLC)

The SDLC is the process of determining how an information system (IS) can support business needs, designing the system, building it, and delivering it to users. Every organization uses a slightly different life-cycle model with several phases, with anywhere from three to almost twenty identifiable phases.

In any given SDLC phase, the project can return to an earlier phase, if necessary. In the systems development life cycle, it is also possible to complete some activities in one phase in parallel with some activities of another phase. Sometimes the life cycle is iterative; that is, phases are repeated as required until an acceptable system is found. Some systems analysts consider the life cycle to be a spiral, constantly cycling through the phases at different levels of detail, as illustrated in below figure.

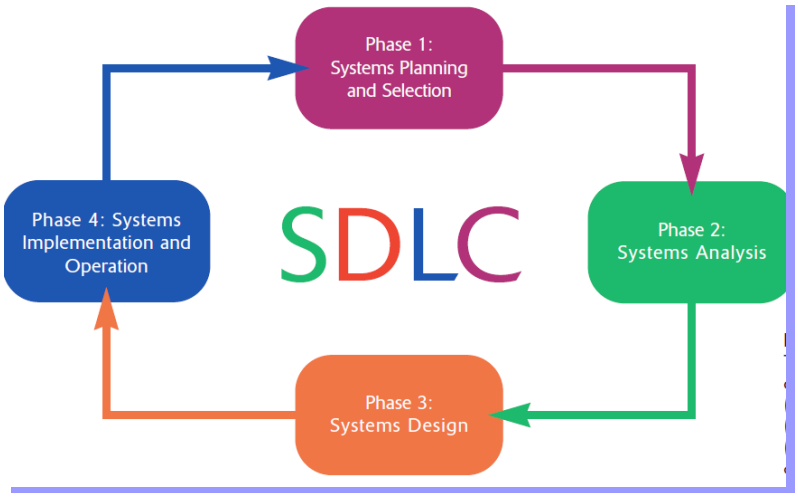


Figure: SDLC with four phases (1) system planning (2) system analysis, (3) system design, and (4) system implementation and operation.

Phase 1: Systems Planning and Selection

- The planning phase is the fundamental process of understanding why an information system should be built and determining how the project team will go about building it.
- The purpose of this phase is to perform a preliminary investigation to evaluate an IT-related business opportunity or problem. A key part of the preliminary investigation is a feasibility study. The feasibility analysis examines key aspects of the proposed project:
  - ✓ The technical feasibility (Can we build it?)
  - ✓ The economic feasibility (Will it provide business value?)
  - ✓ The organizational feasibility (If we build it, will it be used?)

Phase 2: Systems Analysis

- The analysis phase answers the questions of *who* will use the system, *what* the system will do, and *where* and *when* it will be used.
- During this phase, the project team investigates any current system(s), identifies improvement opportunities, and develops a concept for the new system. This phase has three steps:

1. An *analysis strategy* is developed. Such a strategy usually includes a study of the current system (called the as-is system) and its problems, and envisioning ways to design a new system (called the to-be system).
2. The next step is *requirements gathering* (e.g., through interviews, group workshops, or questionnaires). The system concept is then used as a basis to develop a set of business analysis models that describes how the business will operate if the new system were developed.
3. The analyses, system concept, and models are combined into a document called the system proposal, which is presented to the project sponsor and other key decision makers (e.g., members of the approval committee) who will decide whether the project should continue to move forward.

Phase 3: Systems Design

- The *design phase* decides *how* the system will operate in terms of the hardware, software, and network infrastructure that will be in place. The design phase has four steps:

1. A *design strategy* must be determined. This clarifies whether the system will be developed by the company's own programmers, whether its development will be outsourced to another firm, or whether the company will buy an existing software package.
2. This leads to the development of the basic *architecture design* for the system that describes the hardware, software, and network infrastructure that will be used. The *interface design* specifies how the users will move through the system (e.g., by navigation methods such as menus and on-screen buttons).
3. The database and file specifications are developed. These define exactly what data will be stored and where they will be stored.
4. The analyst team develops the program design, which defines the programs that need to be written and exactly what each program will do.

Phase 4: Systems Implementation and Operation

- The final phase of the SDLC is a two-step process: systems implementation and operation.

(a) Implementation includes coding, testing, and installation.
- During coding, programmers write the programs that make up the system.
- During testing, programmers and analysts test individual programs and the entire system in order to find and correct errors.
- During installation, the new system becomes a part of the daily activities of the organization. There are four widely used system conversion strategies i.e., Parallel, Direct, Phased, and Pilot.
- Systems implementation activities also include initial user support such as the finalization of documentation, training programs, and ongoing user assistance.

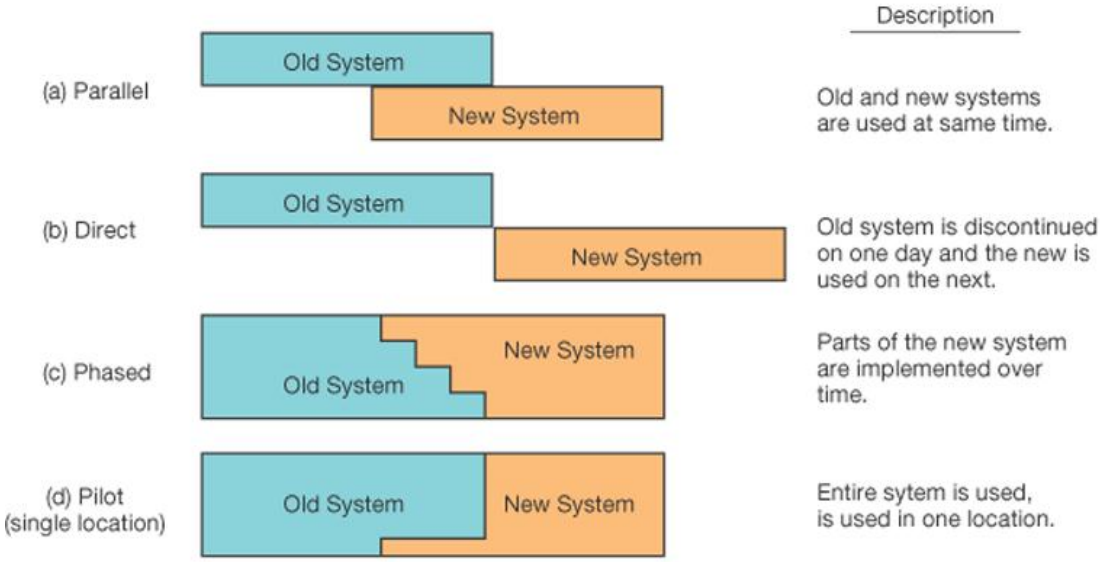
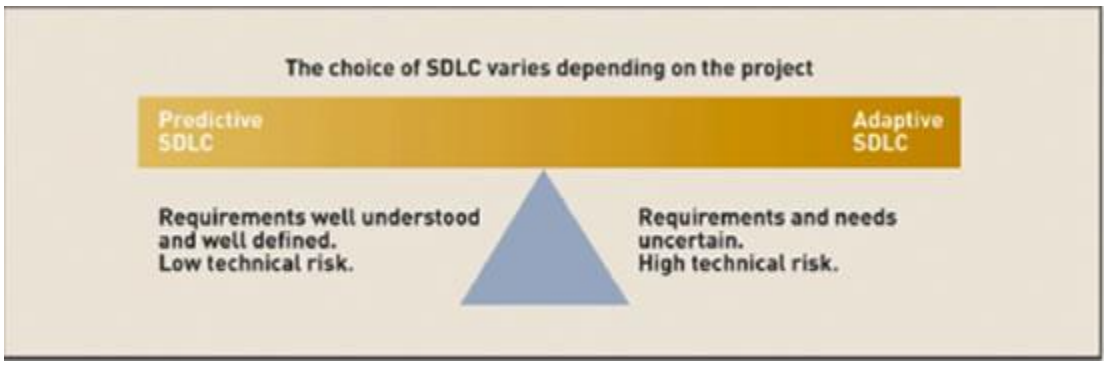


Figure: Software conversion strategy

(b) The second part of the fourth phase of the SDLC is operation.
- During operation, programmers make the changes that users ask for and modify the system to reflect changing business conditions.

PREDICTIVE SDLC VS ADAPTIVE SDLC



Predictive approach to the SDLC

- An approach that assumes the project can be planned in advance and that the new information system can be developed according to the plan.
- Predictive SDLCs are useful for building systems that are well understood and defined. For example, a company may want to convert its old networked client/server system to a newer Web-based system that includes a smartphone app. In this type of project, the staff already understands the requirements very well, and no new processes need to be added. Thus, the project can be carefully planned, and the system can be built according to the specifications.
- The predictive approaches are more traditional and were conceived during the 1970s through the 1990s.

Adaptive approach to the SDLC

- Adaptive methods focus on adapting quickly to changing realities.
- An adaptive approach to the SDLC is used when the system's requirements and/or the users' needs aren't well understood. In this situation, the project can't be planned completely. Some system requirements may need to be determined after preliminary development work.
- Adaptive approaches have evolved with object-oriented technology and Web development; they were created during the late-1990s and into the 21st century.
- An adaptive SDLC shows several iterations, including analysis, design, and implementation activities.

Figure: The iteration of system development activities

TRADITIONAL PREDICTIVE APPROACHES TO THE SDLC - WATERFALL MODEL

- The first formal description of the waterfall model is often cited as a 1970 article by Winston W. Royce, although Royce did not use the term waterfall in that article. The earliest use of the term "waterfall" may have been in a 1976 paper by Bell and Thayer.
- Waterfall development has distinct goals for each phase of development. Imagine a waterfall on the cliff of a steep mountain. Once the water has flowed over the edge of the cliff and has begun its journey down the side of the mountain, it cannot turn back. It is the same with waterfall development. Once a phase of development is completed, the development proceeds to the next phase and there is no turning back.
- The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. Here, each phase must be completed before the next phase can begin and there is no overlapping in the phases.
- This type of model is basically used for the project which is small and there are no uncertain requirements. In this model the testing starts only after the development is complete.
- In Royce's original waterfall model, the following phases are followed in order:
    - System and software requirements: captured in a product requirements document
    - Analysis: resulting in models, schema, and business rules
    - Design: resulting in the software architecture
    - Coding: the development, proving, and integration of software
    - Testing: the systematic discovery and debugging of defects
    - Operations: the installation, migration, support, and maintenance of complete systems

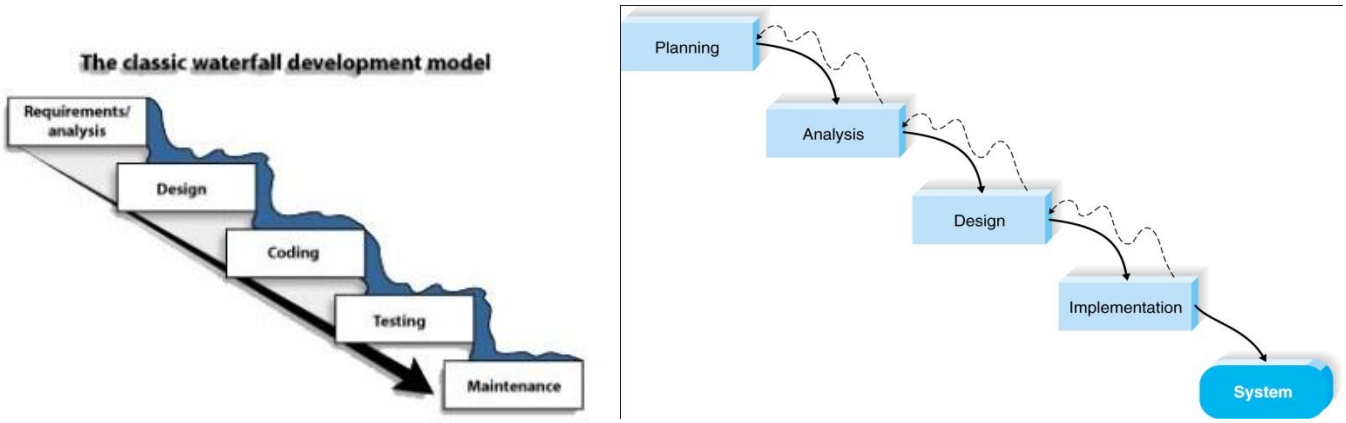


Figure: Waterfall Development

Disadvantages of waterfall model:

- Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Limited user involvement (only in requirements phase)

Variants of Waterfall Development:

There are two important variants of waterfall development.

(a) parallel development
(b) V-model

Parallel development

- Parallel development reduces the time required to deliver a system.
- Here, instead of doing the design and implementation in sequence, a general design for the whole system is performed. Then the project is divided into a series of subprojects that can be designed and implemented in parallel. Once all subprojects are complete, there is a final integration of the separate pieces, and the system is delivered.
- It also adds a new problem: If the subprojects are not completely independent, design decisions in one subproject may affect another, and at the project end, integrating the subprojects may be quite challenging.

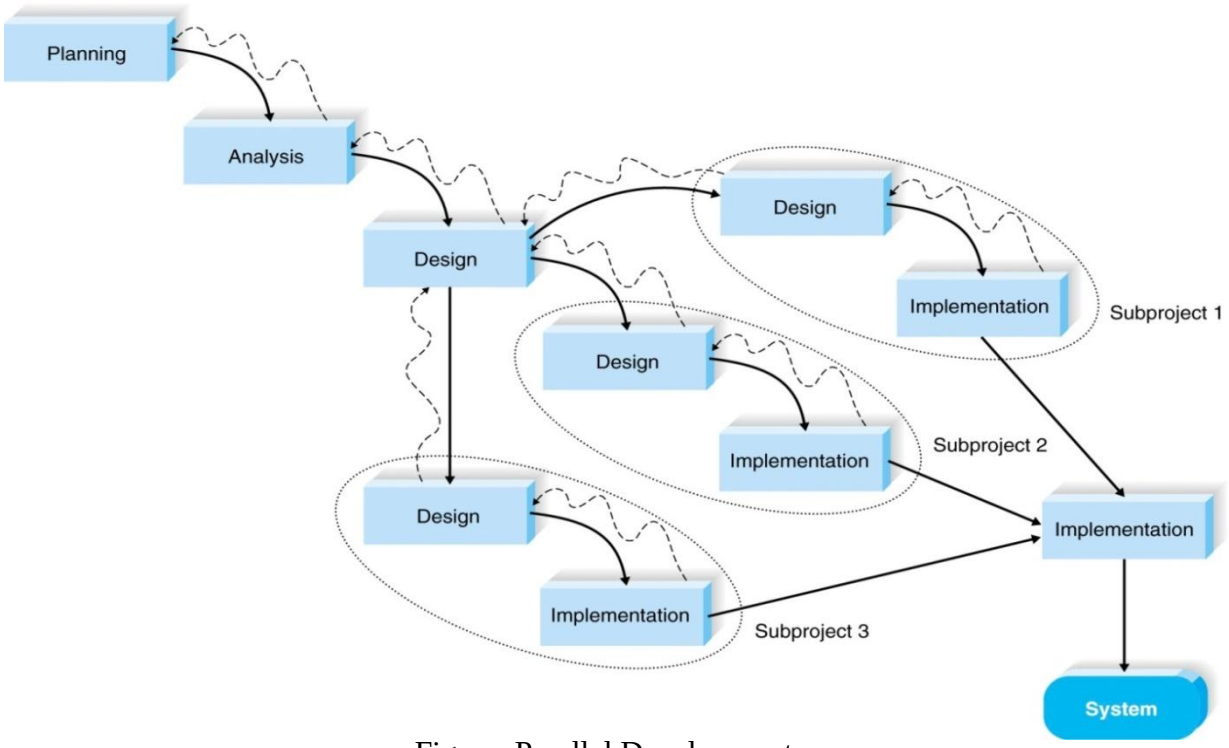


Figure: Parallel Development

V-model:

- V-model means Verification and Validation model. The V-model is another variation of waterfall development that pays more explicit attention to testing.
- 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, unit testing, integration testing, system testing and, finally, acceptance testing are performed.

- 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.
- It still suffers from the rigidity of the waterfall development process.

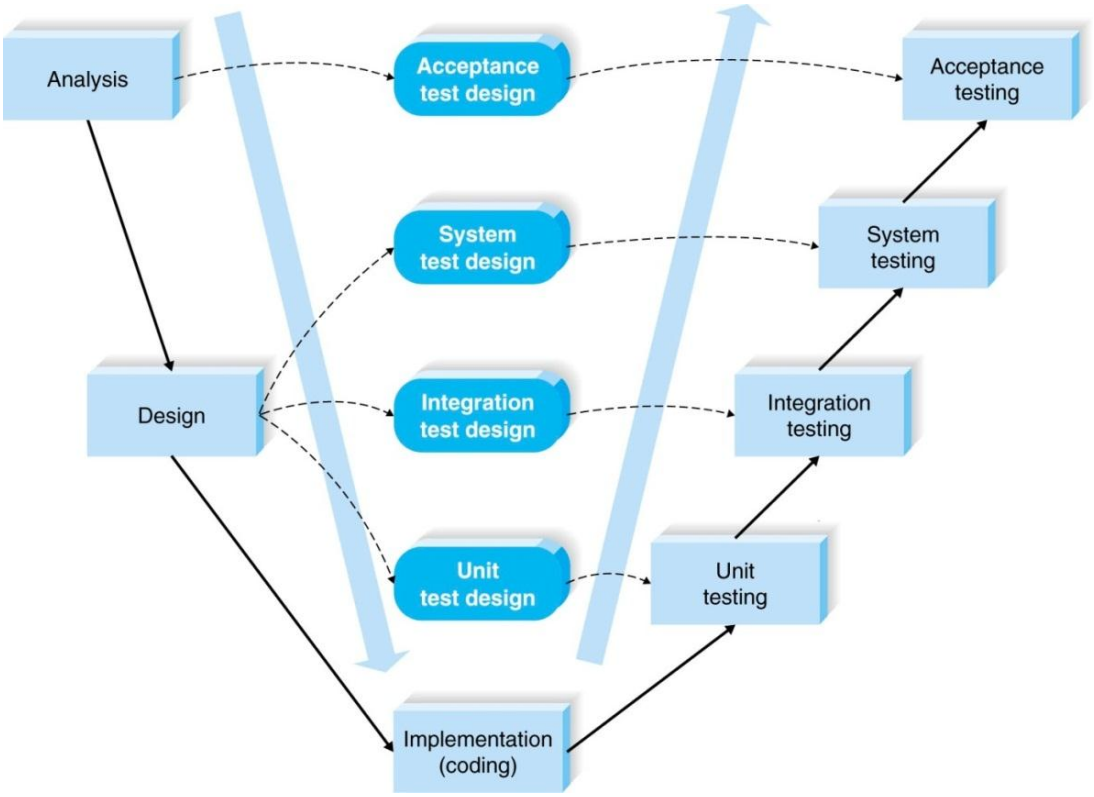


Figure: V-Model

TRADITIONAL ADAPTIVE APPROACHES TO THE SDLC - SPIRAL MODEL

Spiral models initially were suggested in the 1990s by Barry Boehm - a software engineering professor. The life cycle is shown as a spiral, starting in the center and working its way outward, over and over again, until the project is complete. He stated that each iteration, or phase, of the model must have a specific goal that is accepted, rejected, or changed by the user, or client. Thus, each iteration produces feedback and enhancements, which enable the team to reach the overall project goal. Typically, each iteration in a spiral model includes planning, risk analysis, engineering, and evaluation, as shown in the below figure. The repeated iterations produce a series of prototypes, which evolve into the finished system. Proponents of the spiral model believe that this approach reduces risks and speeds up software development.

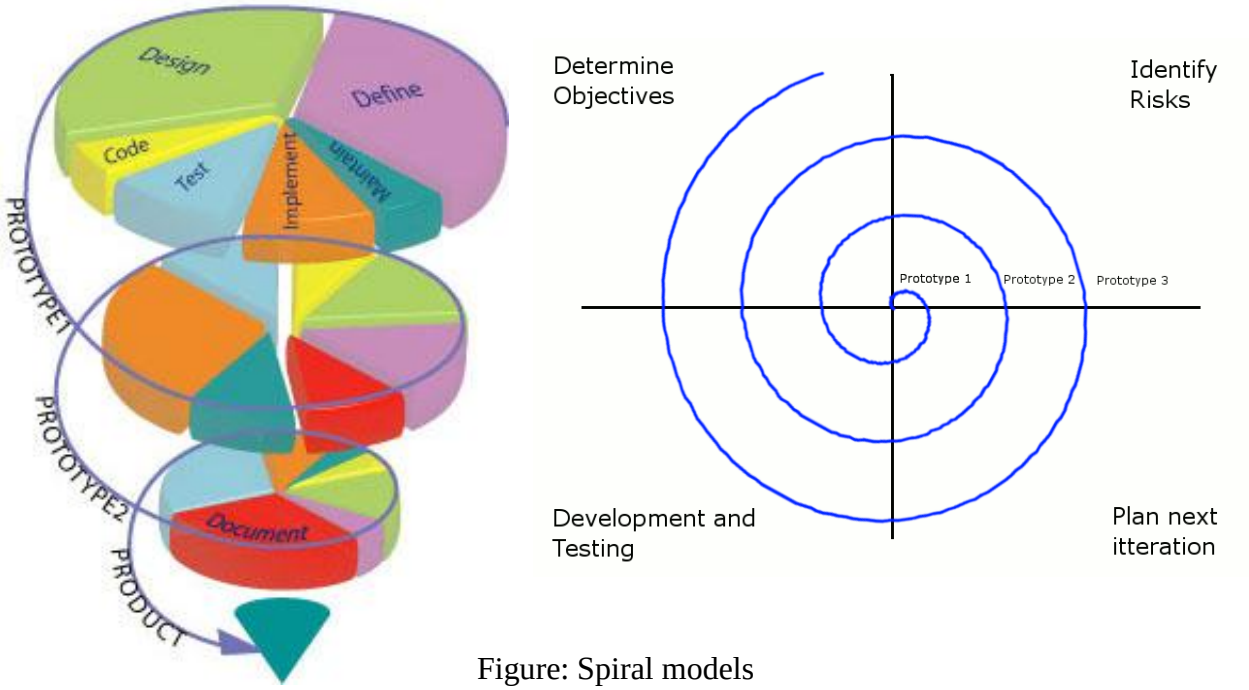


Figure: Spiral models

ALTERNATIVES TO TRADITIONAL WATERFALL SDLC

Prototyping, computer-aided software engineering (CASE) tools, joint application design (JAD), rapid application development (RAD), participatory design (PD), and the use of Agile Methodologies represent different approaches that streamline and improve the systems analysis and design process from different perspectives.

Prototyping

Define:

A prototype is an early working version of an information system. Prototyping tests system concepts and provides an opportunity to examine input, output, and user interfaces before final decisions are made. A prototype can be developed with a CASE tool, a software product that automates steps in the systems development life cycle.

How does it work?

In prototyping, the analyst works with users to determine the initial or basic requirements for the system. The analyst then quickly builds a prototype. When the prototype is completed, the users work with it and tell the analyst what they like and do not like about it. The analyst uses this feedback to improve the prototype and takes the new version back to the users. This iterative process continues until the users are relatively satisfied with what they have seen.

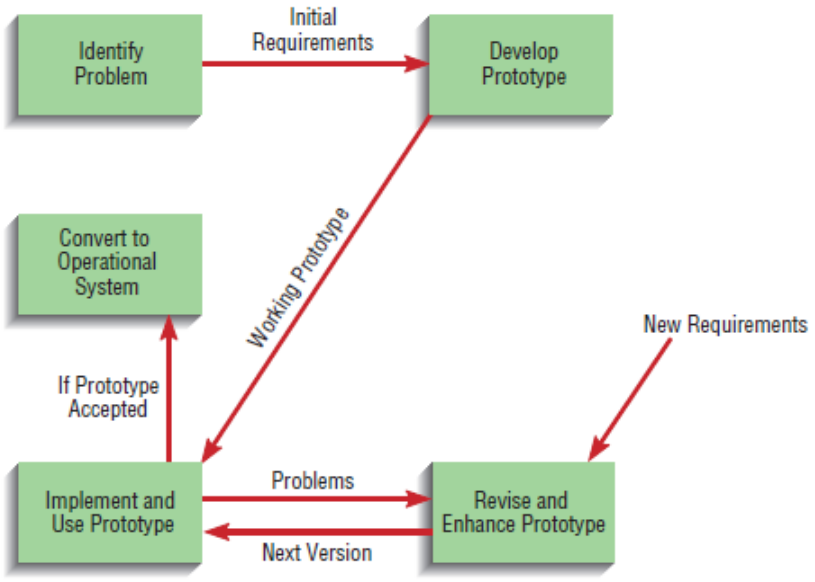


Figure: The prototyping method.

Advantages of Prototyping:

- ✓ Prototyping is most useful for requirements determination when user requirements are not clear or well understood.
- ✓ There is a possibility of developing a system that more closely addresses users' needs and expectations.

Disadvantages of Prototyping:

- ✓ It can be quite difficult to manage prototyping as a project in the larger systems effort.
- ✓ Decisions might be made too early, before business or IT issues are understood thoroughly.

Computer-aided software engineering (CASE):

- Computer-aided software engineering (CASE) is software tools that assist systems builders in managing the complexities of information system projects and helps ensure that high-quality systems are constructed on time and within budget. .
- Visible Analyst (software) is one example of a CASE tool that helps to do graphical planning, analysis, and design in order to build complex client/server applications and databases. Microsoft Visio is a software that also allow users to draw and modify diagrams easily.
- The general types of CASE tools include:
  - ✓ Diagramming tools that enable system process, data, and control structures to be represented graphically.
  - ✓ Display (or form) and report generators make it easier for the systems analyst to identify data requirements and relationships.
  - ✓ A central repository that enables the integrated storage of specifications, diagrams, reports, and project management information.
  - ✓ Documentation generators that help produce both technical and user documentation in standard formats.
  - ✓ Code generators that enable the automatic generation of program and database definition code directly from the design documents, diagrams, forms, and reports.

Joint Application Design (JAD)

- Joint application development (JAD) is a structured process in which users, managers, and analysts work together for several days in a series of intensive meetings to specify or review system requirements.
- It was developed by IBM in the late 1970s.
- The motivation for using JAD is to cut the time (and hence the cost) required by personal interviews, to improve the quality of the results of information requirements assessment, and to create more user identification with new information systems as a result of the participative.
- The following is a list of typical JAD participants:

JAD participant	*Role*
JAD project leader	Develops an agenda, acts as a facilitator, and leads the JAD session
Top management	Provides enterprise-level authorization and support for the project
Managers	Provide department-level support for the project and understanding of how the project must support business functions and requirements
Users	Provide operational-level input on current operations, desired changes, input and output requirements, user interface issues, and how the project will support day-to-day tasks
Systems analysts and other IT staff members	Provide technical assistance and resources for JAD team members on issues such as security, backup, hardware, software, and network capability
Recorder	Documents results of JAD sessions and works with systems analysts to build system models and develop CASE tool documentation

Figure: A typical room layout for a JAD session.

JAD Advantages and Disadvantages:

- When users participate in the systems development process, they are more likely to feel a sense of ownership in the results, and support for the new system.
- Compared with traditional methods, JAD is more expensive and can be cumbersome if the group is too large relative to the size of the project.

Rapid Application Development (RAD)

- 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.
- RAD is an object-oriented approach to systems development that includes various software tools. 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.
- RAD is a team-based technique that speeds up information systems development and produces a functioning information system. Like JAD, RAD uses a group approach, but goes much further. While the end product of JAD is a requirements model, the end product of RAD is the new information system.
- RAD relies heavily on prototyping and user involvement. Based on user input, the prototype is modified and the interactive process continues until the system is completely developed and users are satisfied.
- RAD Phases and Activities:
  The RAD model consists of four phases: requirements planning, (work with users) design system, construction (build the system), and implementation, as shown in below figure. Notice the continuous interaction between the user design and construction phases.

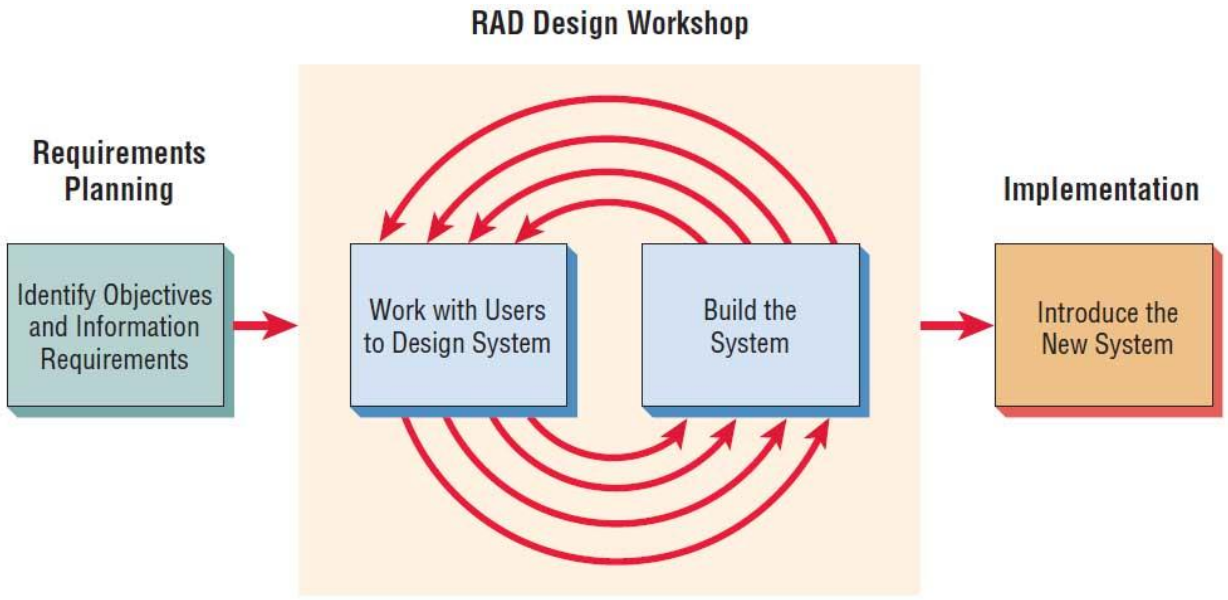


Figure: The four phases of the RAD model

Advantages of RAD:

- The primary advantage is that systems can be developed more quickly with significant cost savings.
- Increases reusability of components
- Encourages customer feedback

Disadvantages of RAD:

- A disadvantage is that RAD stresses the mechanics of the system itself and does not emphasize the company's strategic business needs.
- A system might work well in the short term, but the corporate and long-term objectives for the system might not be met.

Participatory Design

- Participatory design (PD), developed in northern Europe, represents a viable alternative approach to the SDLC.
- PD (also, co-design) is an approach to design attempting to actively involve entire user community - all stakeholders (e.g. employees, partners, customers, citizens, end users) - in the design process to help ensure the result meets their needs and is usable.
- In PD, participants are invited to cooperate with designers, researchers and developers during an innovation process. Potentially, they participate during several stages of an innovation process: they participate during the initial exploration and problem definition both to help define the problem and to focus ideas for solution, and during development, they help evaluate proposed solutions.
- When Participatory Design is suitable:

✓ When we want to better understand how people think about a given problem, discipline or a technology, run a Participatory Design session.

✓ When you feel like there is, or could be, any cultural or political disconnect between you and the end user, a technique such as this might be the best way for you to observe and learn from the user.

Agile Method

- The agile method has evolved in the mid-1990s as a part of reaction against traditional waterfall methods.
- Agile method is a software development method that is people-focused commications-oriented, flexible (ready to adapt to expected change at any time), speedy (encourage rapid and iterative development of the product in small releases), lean (focuses on shortening timeframe and cost and on improved quality), responsive (reacts appropriately to expected and unexpected changes), and learning (focuses on improvement during and after product development).
- Agile method attempts to develop a system incrementally, by building a series of prototypes and constantly adjusting them to user requirements. Agile methods typically use a spiral model.
- Many different individual methodologies come under the umbrella of Agile Methodologies, including the Crystal family of methodologies, Adaptive Software Development, Scrum, Feature Driven Development, RAD (Rapid Application Development) and eXtreme Programming.
- Advantages of Agile Method:

- ✓ Agile is adaptive method and very flexible and efficient in dealing with change. They are popular because they stress team interaction and reflect a set of community-based values.
- ✓ Also, frequent deliverables constantly validate the project and reduce risk.

▪ Disadvantages of Agile Method:

- ✓ Team members need a high level of technical and interpersonal skills.
- ✓ Also, a lack of structure and documentation can introduce risk factors.

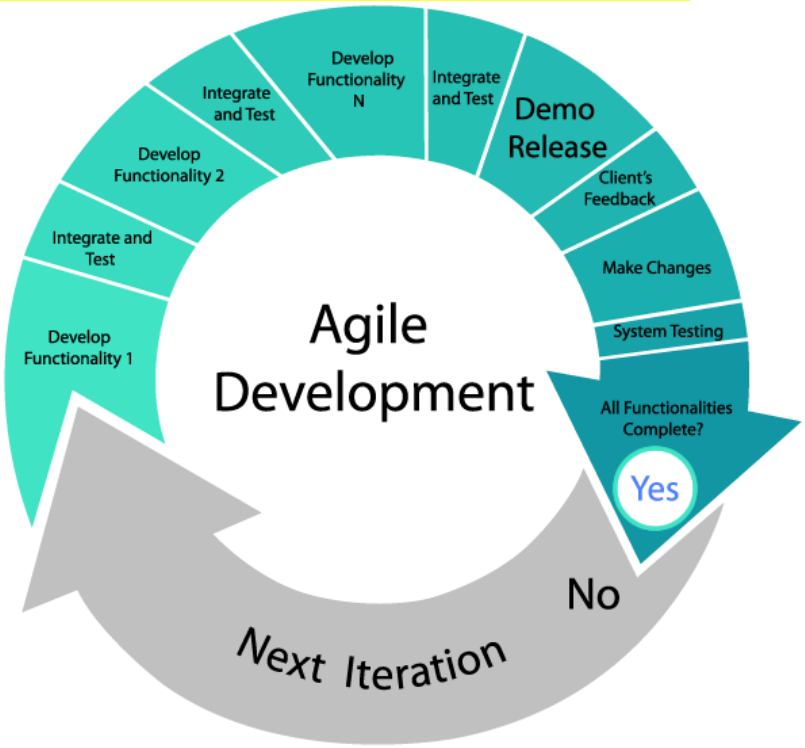


Figure: Agile Methodology

AGILE Vs WATERFALL METHOD

Agile and Waterfall models are two different methods for software development process. Though they are different in their approach, both methods are useful at times, depending on the requirement and the type of the project.

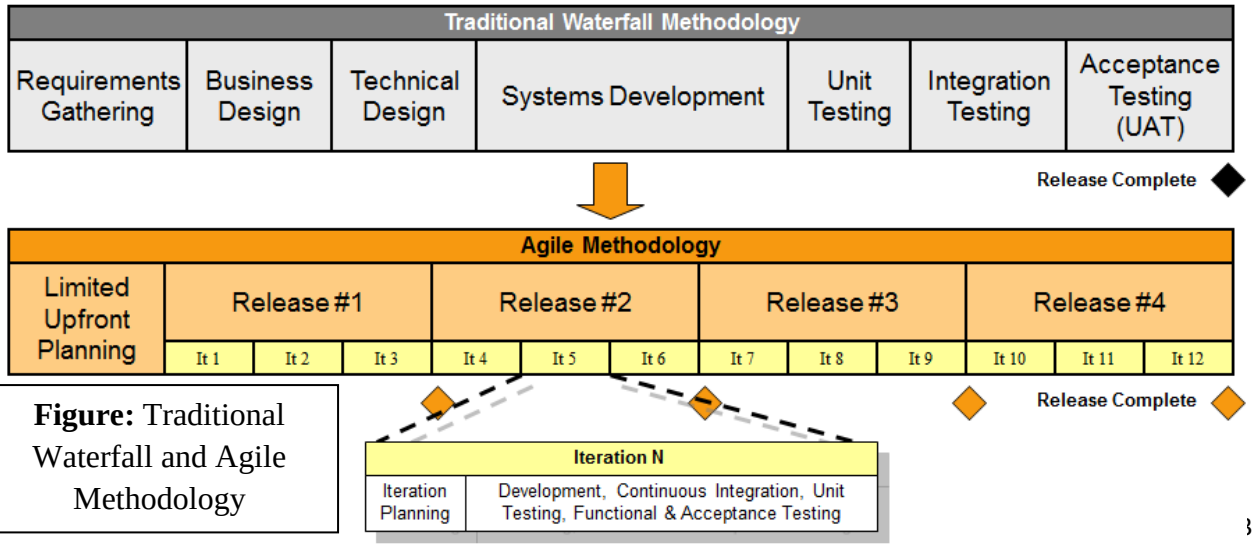


Figure: Traditional Waterfall and Agile Methodology

Agile Model	Waterfall Model
Agile method proposes incremental and iterative approach to software design	Development of the software flows sequentially from start point to end point.
The agile process is broken into individual models that designers work on	The design process is not broken into an individual models
The customer has early and frequent opportunities to look at the product and make decision and changes to the project	The customer can only see the product at the end of the project
Error can be fixed in the middle of the project.	Only at the end, the whole product is tested. If the requirement error is found or any changes have to be made, the project has to start from the beginning
Development process is iterative, and the project is executed in short (2-4) weeks iterations. Planning is very less.	The development process is phased, and the phase is much bigger than iteration. Every phase ends with the detailed description of the next phase.
Documentation attends less priority than software development	Documentation is a top priority and can even use for training staff and upgrade the software with another team
Every iteration has its own testing phase. It allows implementing regression testing every time new functions or logic are released.	Only after the development phase, the testing phase is executed because separate parts are not fully functional.
In agile testing when a product is releases after some iterations, shippable features of the product is delivered to the customer. New features are usable right after shipment. It is useful when you have good contact with customers.	All features developed are delivered at once after the long implementation phase.
Testers and developers work together	Testers work separately from developers
It requires close communication with developers and together analyse requirements and planning	Developer does not involve in requirement and planning process. Usually, time delays between tests and coding

METHODOLOGIES, MODELS, TOOLS, AND TECHNIQUES

Systems developers have a variety of aids at their disposal to help them complete activities and tasks. Among them are methodologies, models, tools, and techniques. The following sections discuss each of these aids.

Methodologies

A system development methodology provides guidelines for every facet of the systems development life cycle. It provides a set of comprehensive guidelines for the SDLC that includes specific models, tools, and techniques.

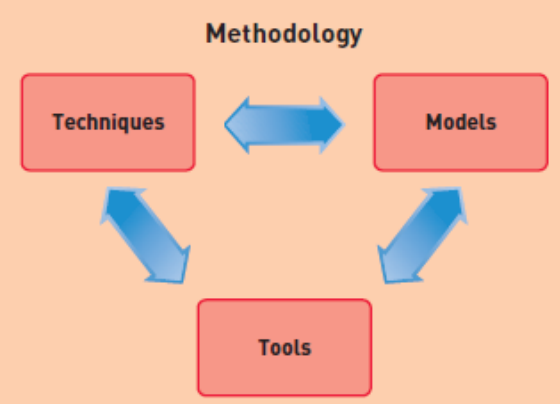


Figure: Components of a methodology

The above figure illustrates that the techniques, models, and tools support one another to provide a comprehensive, integrated methodology.

Models

- The models used in system development include representations of inputs, outputs, processes, data, objects, object interactions, locations, networks, and devices, among other things. Most of the models are graphical models, which are drawn representations that employ agreed-upon symbols and conventions.
- A *business model*, or *requirements model*, describes the information that a system must provide. A *data model* describes data structures and design. An *object model* describes objects, which combine data and processes. A *network model* describes the design and protocols of telecommunications links. A *process model* describes the logic that programmer use to write code modules. Although the models might appear to overlap, they actually work together to describe the same environment from different points of view.
- Lists some models used in system development.
  (a) Some models of system components
    - ✓ Flowchart
    - ✓ Data flow diagram (DFD)
    - ✓ Entity-relationship diagram (ERD)
    - ✓ Use case diagram
    - ✓ Class diagram

  (b) Some models used to manage the development process
    - ✓ Gantt chart
    - ✓ Organizational hierarchy chart
    - ✓ Financial analysis models - NPV, payback period

Tools

- A tool is software support for system developers that helps create models or other components required in the project.
- A project management software tool, such as Microsoft Project, is an example of a tool used to create models.
- Lists the types of tools used in system development:

- ✓ Drawing/graphics application
- ✓ Word processor/text editor
- ✓ Visual modeling tool
- ✓ Integrated development environment (IDE)
- ✓ Database management application
- ✓ Reverse-engineering tool
- ✓ Code generator tool

Technique

- The techniques indicate how the project team will do its work.
- Lists some techniques commonly used in system development.
  - ✓ Project management techniques
  - ✓ User interviewing techniques
  - ✓ Relational database design techniques
  - ✓ Structured programming technique
  - ✓ Software-testing techniques
  - ✓ User-interface design techniques

APPROACHES (METHODOLOGIES) TO SOFTWARE DEVELOPMENT

All system developers should be familiar with two very general approaches to software construction and modeling because these form the basis of virtually all methodologies: the structured approach and the object-oriented approach.

(A) The Structured Approach

Structured analysis, structured design, and structured programming are the three techniques that make up the structured approach are collectively referred to as the structured analysis and design technique (SADT).

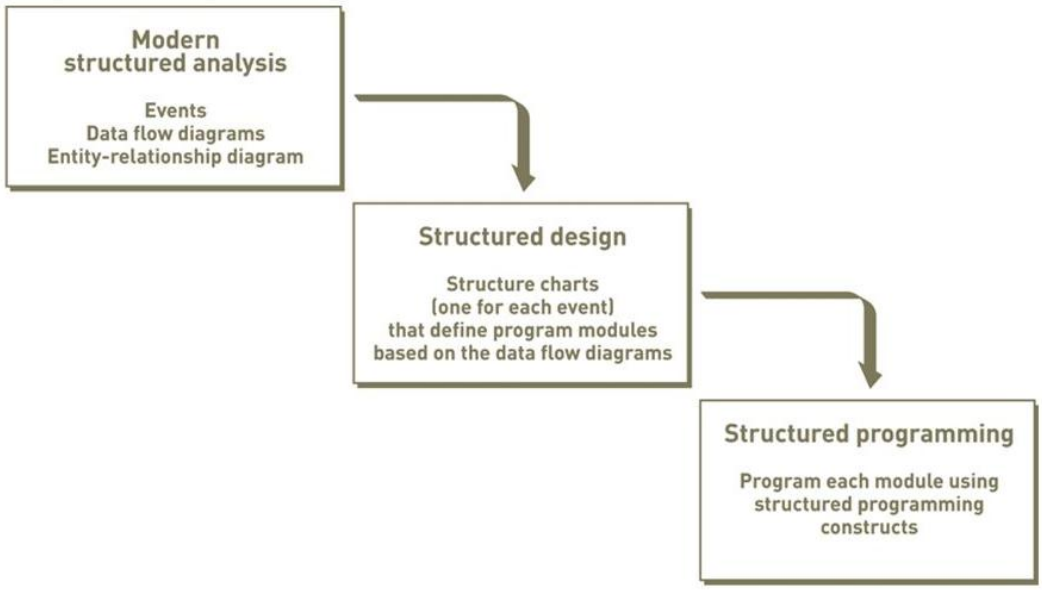


Figure: How structured analysis leads to structured design and structured programming

Structured Programming:

Structured programming produces a program that has one beginning and one ending, with each step in the program execution consisting of one of three programming constructs:

✓ A sequence of program statements
✓ A decision point at which one set or another set of statements executes
✓ A repetition of a set of statements

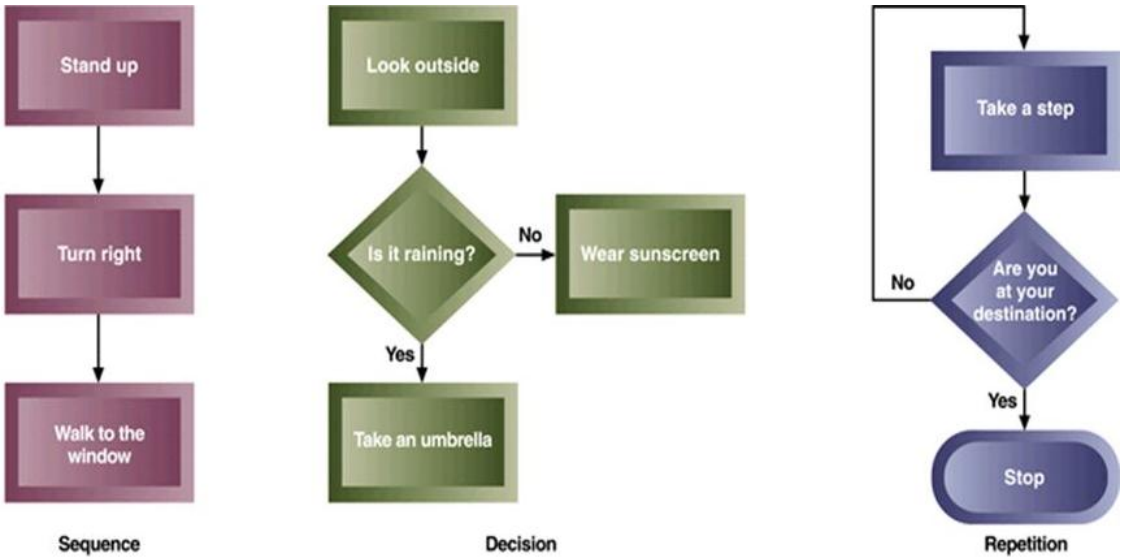
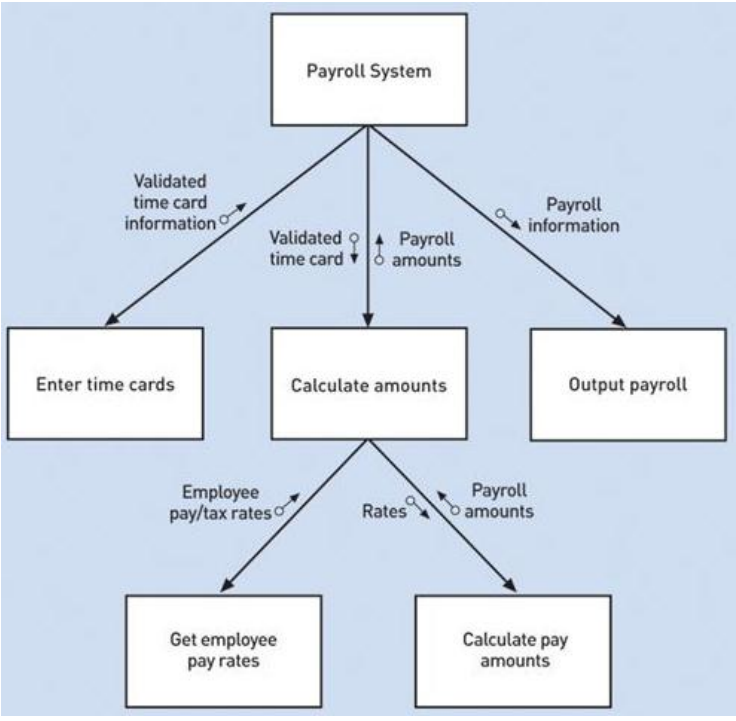


Figure: Three structured programming constructs

Structured Design:

Figure: A structure chart created by using the structured design technique.



The structured design technique was developed to provide some guidelines for deciding what the set of programs should be, what each program should accomplish, and how the programs should be organized into a hierarchy. The modules and the arrangement of modules are shown graphically by using a model called a structure chart (a graphical diagram showing the hierarchical organization of modules).

Structured Analysis

The structured analysis technique helps the developer define what the system needs to do (the processing requirements), what data the system needs to store and use (data requirements), what inputs and outputs are needed, and how the functions work together to accomplish tasks. The key graphical model of the system requirements that are used with structured analysis is called the data flow diagram (DFD); it shows inputs, processes, storage, and outputs as well as the way these function together.

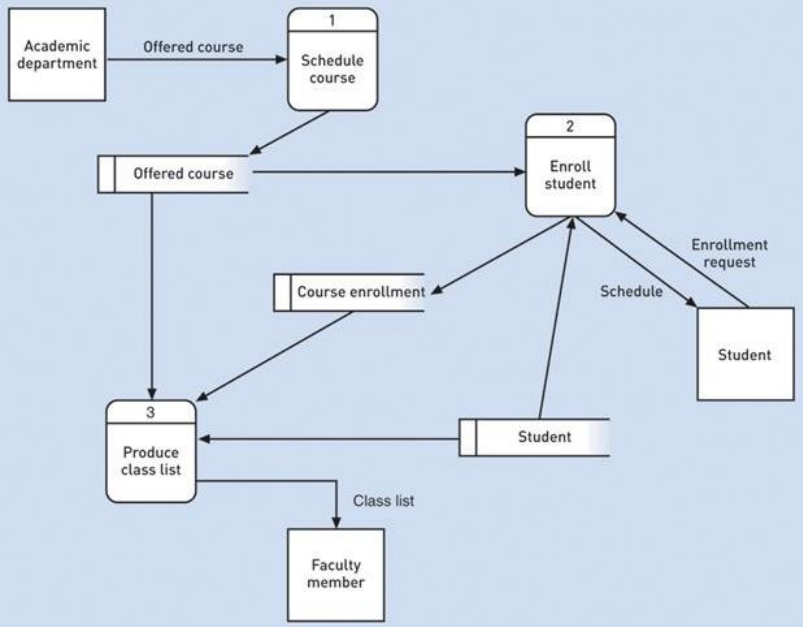


Figure: A data flow diagram (DFD) created by using the structured analysis technique

(B) Object-oriented approach

- ✓ The object-oriented approach views an information system as a collection of interacting objects that work together to accomplish tasks.
- ✓ Conceptually, there are no processes or programs; there are no data entities or files. The system consists of objects (An object is a thing in the computer system that is capable of responding to messages).
- ✓ Code is modular and reusable, which can reduce cost and development time.
- ✓ Easy to maintain and expand as new objects can be cloned using inherited properties
- ✓ More recent object-oriented languages include C++, Java, and C#. These languages focus on writing definitions of the types of objects needed in a system; as a result, all parts of a system can be thought of as objects, not just the graphical user interface.

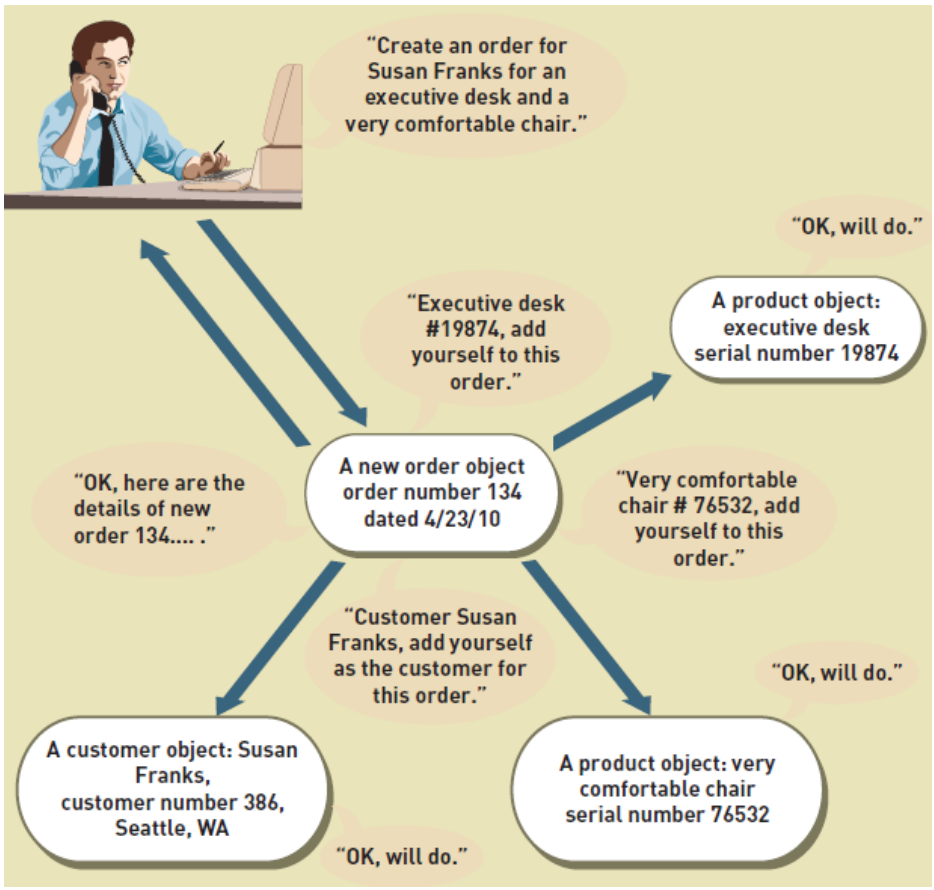


Figure: The object-oriented approach to information systems (read clockwise starting with the user)

A computer system requires a different approach to systems analysis, systems design, and programming.

Object-oriented analysis (OOA)

OOA defines the objects that do the work and determines what user interactions (called use cases) are required to complete the tasks.

Object-oriented design (OOD)

OOD defines all the additional types of objects that are necessary to communicate with people and devices in the system, it shows how the objects interact to complete tasks, and it refines the definition of each type of object so it can be implemented with a specific language or environment.

Object-oriented programming (OOP)

OOP is the writing of statements in a programming language to define what each type of object does.

CHOOSING WHICH SYSTEMS DEVELOPMENT METHOD TO USE

Choosing a methodology is not simple, because no one methodology is always best.

Usefulness in Developing System	Waterfall	Parallel Waterfall	V-Model Waterfall	Prototyping	RAD	Agile Development
with unclear user requirements	Poor	Poor	Poor	Excellent	Excellent	Excellent
that are complex	Good	Good	Good	Poor	Good	Poor
that are reliable	Good	Good	Excellent	Poor	Good	Good
with short time schedule	Poor	Good	Poor	Excellent	Excellent	Excellent
with schedule visibility	Poor	Poor	Poor	Excellent	Good	Good

INCREMENTAL DEVELOPMENT VS ITERATIVE DEVELOPMENT

The terms *iteration* and *increment* are often used freely and interchangeably.

Incremental Development

- Incremental model are broken down into multiple standalone modules of software development cycle.
- Incremental development is always based on an iterative life cycle. The basic idea is that the system is built in small increments. An increment may be developed within a single iteration or it may require two or three iterations. As each increment is completed, it is integrated with the whole.
- Each iteration passes through the requirements, design, coding and testing phases. And each subsequent release of the system adds function to the previous release until all designed functionality has been implemented.

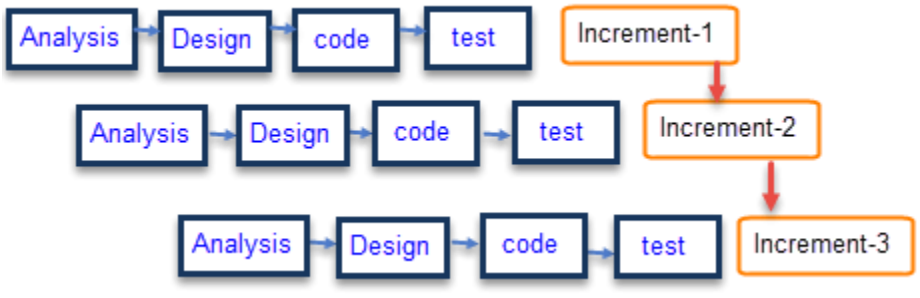


Figure: Incremental model

Iterative Development

- Iteration refers to the cyclic nature of a process in which activities are repeated in a structured manner.
- Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added.
- The key to successful use of an iterative software development life cycle is rigorous validation of requirements, and verification (including testing) of each version of the software against those requirements within each cycle of the model.

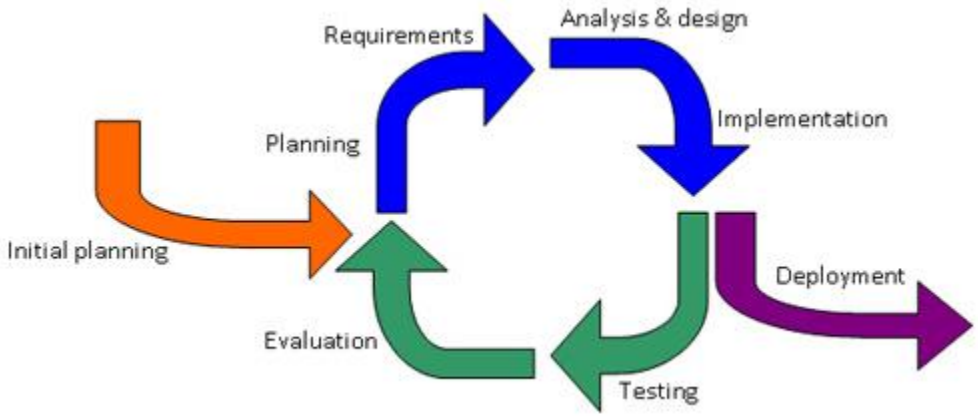


Figure: Typical iterative development process

Real life example:

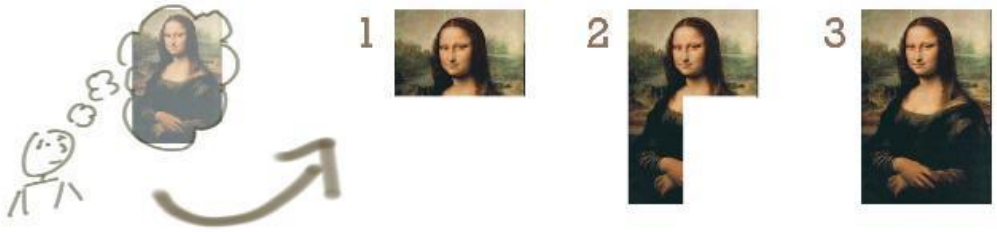


Figure: Incremental Development

Figure: Iterative Development

SYSTEMS DEVELOPMENT GUIDELINES

The basic principles in the below table apply to any IT project, large or small. These basic guidelines apply throughout the systems development process. Although you will develop your own methods and techniques, these guidelines will help you achieve success as a systems analyst.

Table: Five Basic Systems Development Guidelines

Develop a Plan	Prepare an overall project plan and stick to it. Complete the tasks in a logical sequence. Develop a clear set of ground rules and be sure that everyone on the team understands them clearly.
Involve Users and Listen Carefully to Them	Ensure that users are involved in the development process, especially when identifying and modeling system requirements. When you interact with users, listen closely to what they are saying.
Use Project Management Tools and Techniques	Try to keep the project on track and avoid surprises. Create a reasonable number of checkpoints — too many can be burdensome, but too few will not provide adequate control. Try to use Microsoft Project to help you manage tasks, allocate resources, and monitor progress.
Develop Accurate Cost and Benefit Information	Managers need to know the cost of developing and operating a system, and the value of the benefits it will provide. You must provide accurate, realistic cost and benefit estimates, and update them as necessary.
Remain Flexible	Be flexible within the framework of your plan. Systems development is a dynamic process, and overlap often exists among tasks. The ability to react quickly is especially important when you are working on a system that must be developed rapidly.