2nd Lecture
Methodologies for information system development

- Concepts used in computer systems development
- Methodologies – definition and content
- Classification of computer system development methodologies
- Structures methodologies:
  - SSADM
  - MERISE
- Object-oriented methodologies
  - OMT (Object Modeling Technique)
  - Iterative development of computer systems using UML
  - Unified methodology for computer system development (RUP)
- Methodologies based on rapid development
- Methodologies based on agile development

Concepts used in computer system development

Concept	Definition
Process/stage	It is a **set of interrelated activities** that use resources to achieve a well-established **objective**. There can be primary, support and management processes. Some methodologies use the name "**path**" or "**workflow**" for the same concept.
Activity	It includes the **action types** carried on for efficient use of resources. It is part of a process. Some methodologies use the term **phase**, **step** or **segment** for the same concept
Phase	It represents **the time between two key points** of a process, while a well-defined set of objectives is achieved in Rational Unified Process.
Step	It represents a **sequence of activities** carried on in a work stage – in SSADM

Concepte utilizate în realizarea sistemelor informatice

Concept	Definition
Tasks	They are **components of activities** and are a **set of actions** that constitutes ***the responsibility of a persons*** or ***group of persons***. A task has preconditions, deliverables and postconditions. The succession in time of tasks, activities, stages and processes form **the lifecycle of the computer system.**
Lifecycle of a computer system	It is defined by the succession in time of tasks, activities, stages and processes. It is a **template for ordering the activities** for computer system development, covering the **time interval** that begins with the decision to develop an information system and finishes with the decision to abandon it and replace it with a new one.
Development cycle of a computer system	It is contained in the lifecycle of the computer system. It includes the time from making the decision of developing a computer system until the system is put into use.

Methodologies: concepts and definitions

A methodology for computer system development should include:

- **Stages/processes** of development structured in sub-stages, activities, tasks and their content;
- **The flow** of implementation for these stages/ processes, sub-stages and activities;
- **The way of running computer system life-cycle;**
- Computer system **approach;**
- **Implementation strategies and methods;**
- **Rules of formalization** for the components of the computer system;
- The used **techniques, procedures, tools, regulations** and **standards**;
- **Project management** activities (planning, scheduling, tracking) and the use of material, financial and human **resources**

Classification of computer system development methodologies

A. By the generality degree:

- ***General methodologies*** have a high degree of generality and can be used for developing computer systems for various fields. For example: SSADM (Structured System Analysis and Design Methodology), MERISE (Méthode d'Etude et de Realization Informatique pour les Systèmes d'Entreprise), OMT (Object Modeling Technique), RUP (Rational Unified Process).

- ***Framework methodologies*** cover elements that can apply to some ***specific software systems.*** For example: Selection and Implementation of Integrated Packaged Software (SIIPS) developed by **KPMG**. It has implementation accelerators for ORACLE and SAP.

- ***Specialized methodologies*** are developed and applied for a ***single software product.*** For example: AIM (for Oracle E-Business Suite), POIS (for Sun Systems), Signature (for Scala), ASAP (for SAP).

Classification of information system development methodologies

B. By computer system approach:

- ***Structured approach methodologies*** – *their work principle is system division in sub-systems by system functions (**function-based approach**) or by processed data (**data-based approach**)*. They **suggest the separation of data modeling from procedure modeling**. Procedure modeling is based on the idea that functions are active and have a behavior, while data is affected by these functions.

- ***Object-oriented methodologies*** they allow computer system development using concepts from ***object-oriented technology***. Object-oriented technology has emerged with the rise of object-oriented programming languages. The first object-oriented languages were: SIMULA (1960), SMALLTALK (1970), CLOS, EIFFEL, ACTOR, C++, Object Pascal (1980).

Structured approach methodologies

The benefits of using structured approach methodologies:

- The use of **graphic representation**, within reach of both analysts and beneficiary;
- Efficient project planning by dividing it in **sub-systems**;
- A well-structured environment allows a **flexible** behavior;
- It has **clear objectives** and a **well-defined scope**;
- It offers the possibility of **cutting down the development time and costs** taking into considerations, from the very beginning, the details of the system and continuously interaction with the beneficiary.
- Changing a certain activity does not lead resumption of the whole study.

Structured approach methodologies

Exemples of structured approach:

- Structured Analysis and Design Information Systems (STRADIS). It is the first described methodology, proposed by Cris Gane and Trish Sarson.
- Yourdon Systems Method (YSM).
- Information Engineering (IE).
- Structured System Analysis and Design Methodology (SSADM).
- Méthode d'Etude et de Realization, Informatique pour les Systèmes d'Entreprise **MERISE** .
- Jackson System Development (JSD) .
- Information System Work and Analysis of Changes (ISAC).
- Effective Techical and Human Implementation of Computer-based Systems (ETHICS) .
- Soft System Methodology (SSM) .
- Multiview
- Process Innovation.
- Rapid Application Development (RAD) .
- ICI methodology - Institutul Centrului de Informatică, România

Object-oriented methodologies

The advantages of object-oriented methodology:

- **Data and processing** are not distinctly represented, but encapsulated in the object class
- **The analysis of the system** can by modified to be **reused** in the analysis of other systems from the same field of activity.
- The used **models** are **flexible** and **easy to maintain**.
- There is the possibility to approach more and more challenging fields and problem types.
- **High consistency** between all the models developed during object-oriented analysis, design and programming.
- **Robustness of the system.**
- **Reuse** of the results of analysis, design and implementation.
- **Explicit representation** of elements that are common to the whole system components.

Object-oriented methodologies

Exemples of object oriented methodologies:

- Object Oriented Software Engineering (**OOSE**) conceived by Ivar Jacobson.
- Object Modeling Technique (**OMT**) elaborated by James Rumbaugh, Michael Blaha et al. The methodology was first used by General Electric and Development Center;
- Object Oriented Design (**OOD**) elaborated by Grady Booch, it is similar to OMT, it focuses on the same idea – iterative analysis and design, insisting on the design step.
- Object Oriented Analysis (**OOA**) elaborated by Peter Coad and Edward Yourdon;
- Object Oriented Structured Design (**OOSD**) elaborated by Wasserman;
- Object Oriented System Analysis (**OOSA**) is a development methodology for **real-time systems**, conceived by Sally Shlaer and Steven Mellor. The authors continued to improve this methodology and published a paper about UML notation applied in Shlaer/Mellor methodology;
- Responsibility Driven Design (**RDD**), was elaborated by Wirfs – Brock, Wilkesson and Wienner;
- Object Oriented Role Analysis, Synthesis and Structuring was elaborated by Reens Kaugh;

Most of the differences between OOD, OAD, OOSA, OMT and OOSE were removed in 1997 by elaborating a standard for notations, diagram types, model types etc, called UML (Unified Modeling Language).

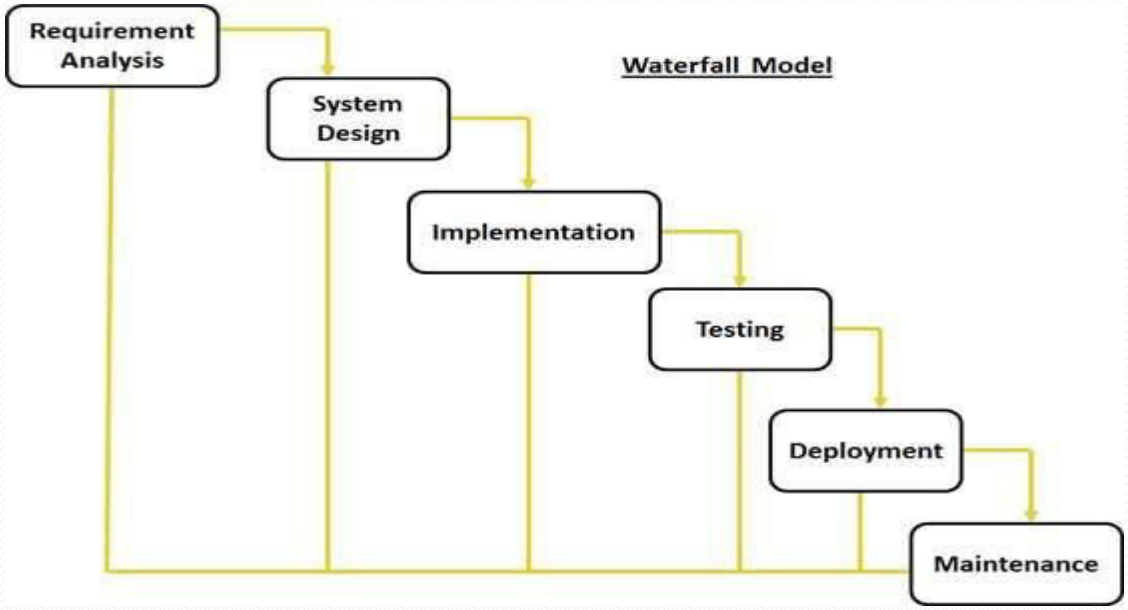
Classification of information system development methodologies

C. By the lifecycle model:

I. *Waterfall lifecycle*
II. *Spiral lifecycle*
III. *Extention based lifecycle*
IV. *Evolutive lifecycle*
V. *Composed lifecycles (V-like cycles and X-like cycles)*

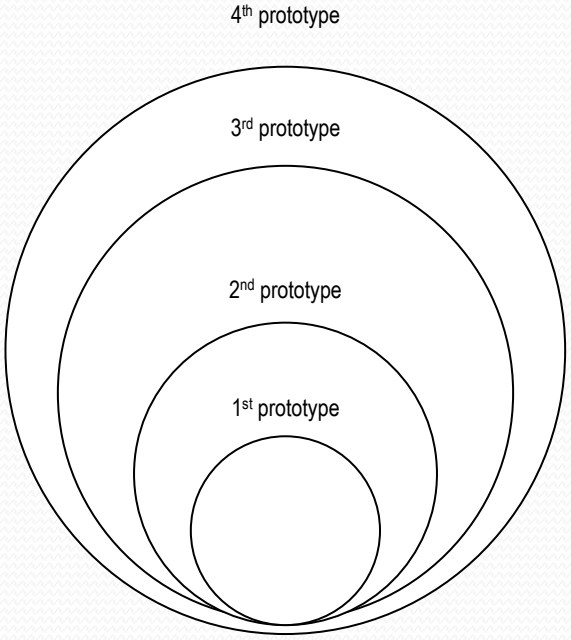
I. The waterfall lifecycle model

- ✓ The development proceeds sequentially, with possible return to the previous stage.
- ✓ It is used for information systems having **low complexity**.
- ✓ The waterfall/linear lifecycle model is a **theoretic model**, as in real-life, going through stages is an iterative process and several activities are often executed in parallel.



II. Spiral lifecycle model (prototype model)

- ✓ It assumes full, **quick**, **low cost development** of an initial, simplified version of the system, that will play the role of a **prototype**. Based on that prototype, we will develop new definition specifications for the computer system and a new version of the system will be achieved
- ✓ The elaboration of the new version involves **full** or **partial** completion of the phases, changing only some only parts of a prototype.



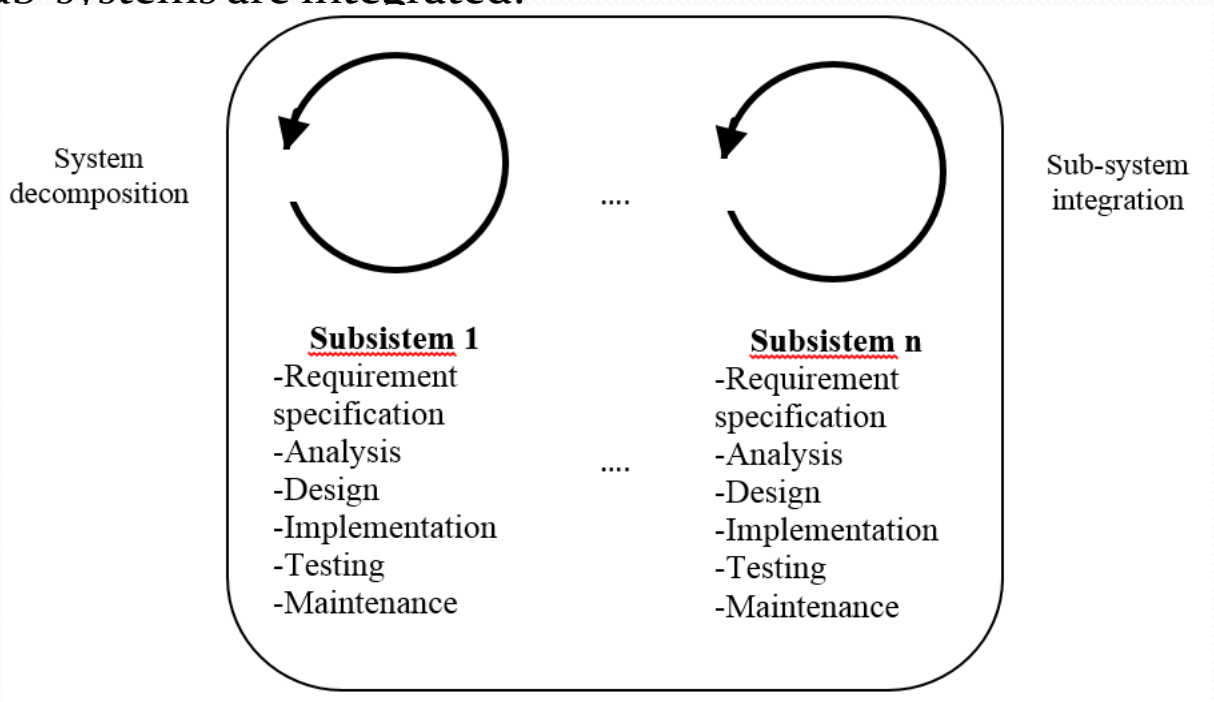
III. Extension lifecycle model (incremental model)

- It is used when the computer systems can be partially developed and put into exploitation, dividing it in **sub-systems**, **applications**, **modules**.

- Their development can be accomplished in an **extensible manner**, it starts with the requirements analysis and definition and then sub-systems are developed and integrated in **successive or simultaneous extensions**
- Typically they branch of from design phase of the system

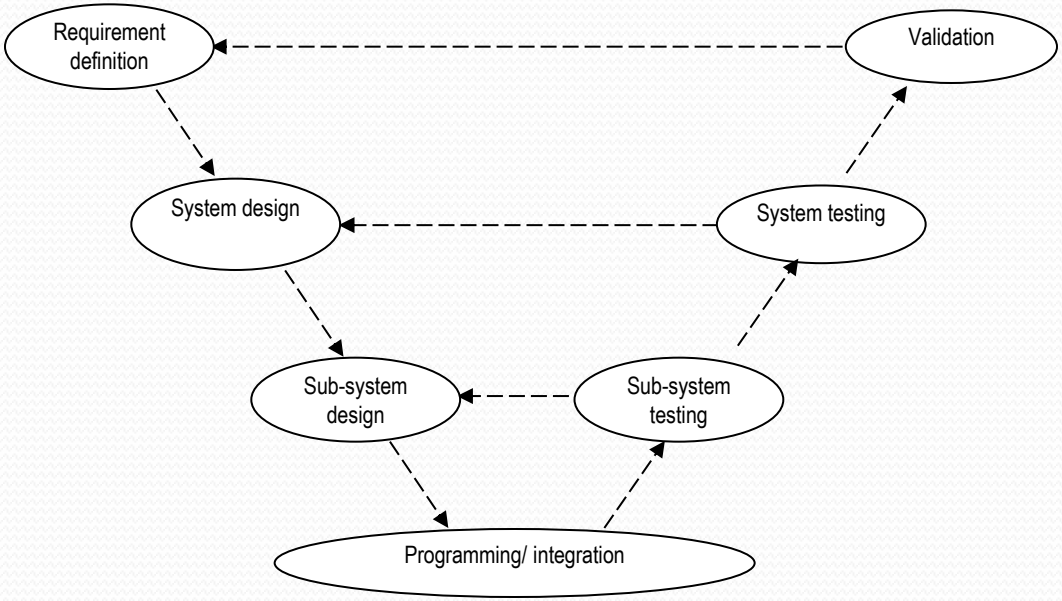
IV. Evolutive lifecycle model

- ✓ It is recommended for **complex** systems that can be **decomposed** din **sub-systems**. They are **iteratively** developed and delivered and contribute to gradually improve the system performances.
- ✓ Each subsystem passes **all the system development phases**: requirement definition, analysis, design, implementation, testing, maintenance. Finally, the sub-systems are integrated.



V. Composite lifecycle model (V-shaped model)

- ✓ It is **a variant of the waterfall model**, that apply **explicit tests** for increasing the control on the way the stages take place
- ✓ The **left side** of the "V" letter is downward passed and it contains the actual development stages and the **right side** of the "V" is passed upward and includes checking and validation of previously created elements.

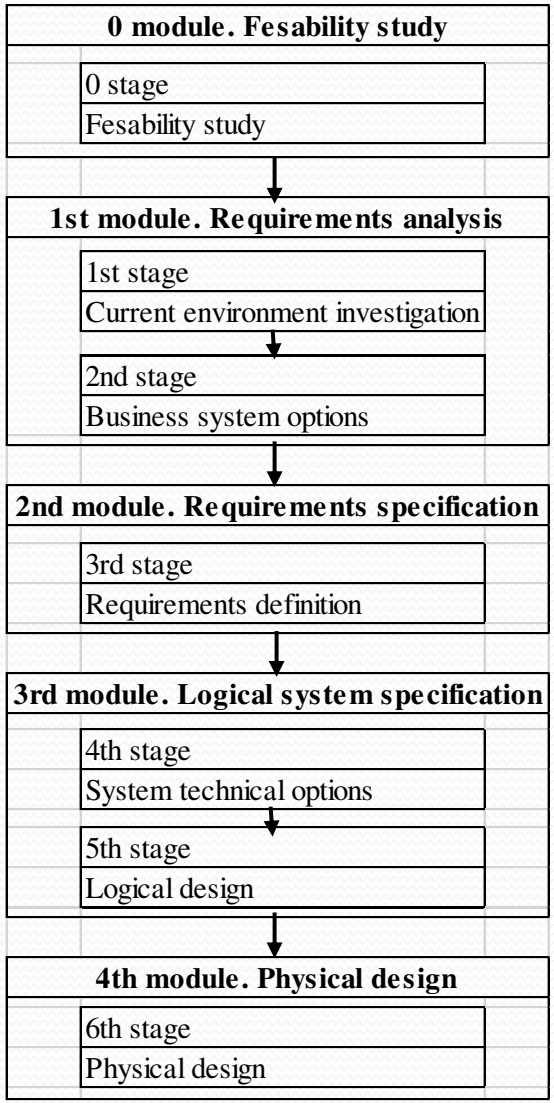


SSADM – system development stages

SSADM includes as set of techniques, tools and standard forms for describing the **existent system** or the **designed system** (new system). General features:

- It is a **data structure** oriented methodology.
- It emphasizes two model types: system ***physical model*** and system ***logical model***, so it separates physical design from logical design
- It is based on clear specification of requirements and of detailed rules for system development (design) of the two models. It uses diagrams for representing data flows and processing.
- It contains **5 modules**: feasibility study, requirements analysis, requirements specification, logical system specification and physical design. Each module is broken down in **stages**. Each stage is further broken down in a number of **steps** that define inputs, outputs and tasks that have to be accomplished.

SSADM – system development stages



MERISE methodology

It identifies three information system cycles: decision cycle, lifecycle, abstracting cycle.

1. **The decision cycle** consists of all the decision mechanisms, including those for choosing options, during the development of the information system. Each decision point during the development of an information system is identified by Merise. It is essential to know who takes the decisions, particularly those relating to the validation of the various models used by the method, and when to complete one stage or start the next.
2. **Information system life cycle** is similar to SSADM, but with the addition of a strategic planning phase, which maps the goals of the organisation to its information needs, and partitions the organisation into "domains" for further analysis. . It has three phases:
    1. **System conception** materialized in system **functional and technical specifications**,
    2. **System realization** materialized in system **detailed specification** and
    3. **System implementation**.
3. **The abstraction cycle** is a gradually descending approach which goes from the knowledge of the problem area (conceptual); to making decisions relating to resources and tasks; through to the technical means on which to implement it. The data view is modeled in three stages:
    - *Conceptual stage* looks at the organisation as a whole;
    - *Organizational stage* for logical level of processing and data
    - *Operational stage* for physical level of processing and data

OMT - Information system development stages

- System modeling is performed based on three different models:
  - **The object model** describes, in terms of static objects, relationships between objects, attributes and operations for each class of objects. It is basically a data model, seen from the perspective of object-oriented approach, showing **WHAT** is analyzed
  - **The dynamic model** highlights data states and the event flow that cause the state change
  - **The functional model** describes how to obtain information outputs from inputs and other intermediate information.
- The phases of the computer system development according to this methodology are:
  1. **Analysis**,
  2. **System design**,
  3. **Object design,**
  4. **Implementation**.

OMT - Information system development stages

Stage	Specific activities
System analysis	○ *Problem definition* ○ ***Object model* initiation** ○ ***Dynamic model initiation*** • identifying inputs and outputs (regarded as parameters of events in the system) and representation of data flow diagram (DFD) • description of elementary processes • identifying constraints • identifying ways to optimize
System design	○ *The decomposition into subsystems* ○ *Identification of concurrent subsystems* ○ *Establishing the necessary resources and how to implement hardware and software for each subsystem* ○ *Choosing the mode of organizing data and types of data access* ○ *Establishing the internal and external control of the event or processing flow.* ○ *Establishing boundary conditions that relate to initialization, normal or abnormal termination, priorities*

OMT - Information system development stages

Stage	Specific activities
Object design	The models obtained in the analysis phase are refined by adding details of implementation: • Identification of operations • Design of algorithms; • Refining, restructuring data model; • Implementation of associations; • Data and association grouping in modules
Implementation	The system is transposes into a programming language and the system in transferred in real environment.

Exemple of iterative process for computer system development with UML (Unified Modeling Language)

Problem identification
It identifies the main characteristics of the studied economic unit and the mode of operation of the implemented activity .
Solution structuring
Beneficiary requirements are detailed. It includes the following steps: **Actor setting; Use case setting; setting relationships between use cases; Use case diagrams construction.**
System analysis
System specifications and use cases are analyzed and there are identified the **main concepts** the system will work with and the relationships between them. The following diagrams are built: **class diagram, object diagrams, state chart diagrams, activity diagrams, sequence diagrams, communication diagrams.**
System design
It has two sub-stages: **architecture design** and **detailed design**. It involves system architecture design, database design, interface design, algorithm design. The following diagrams are built: **component diagrams** and **deployment diagrams**.
System implementation
It involves the actual programming of the identified classes by writing **source code** and implementing **forms and reports**.

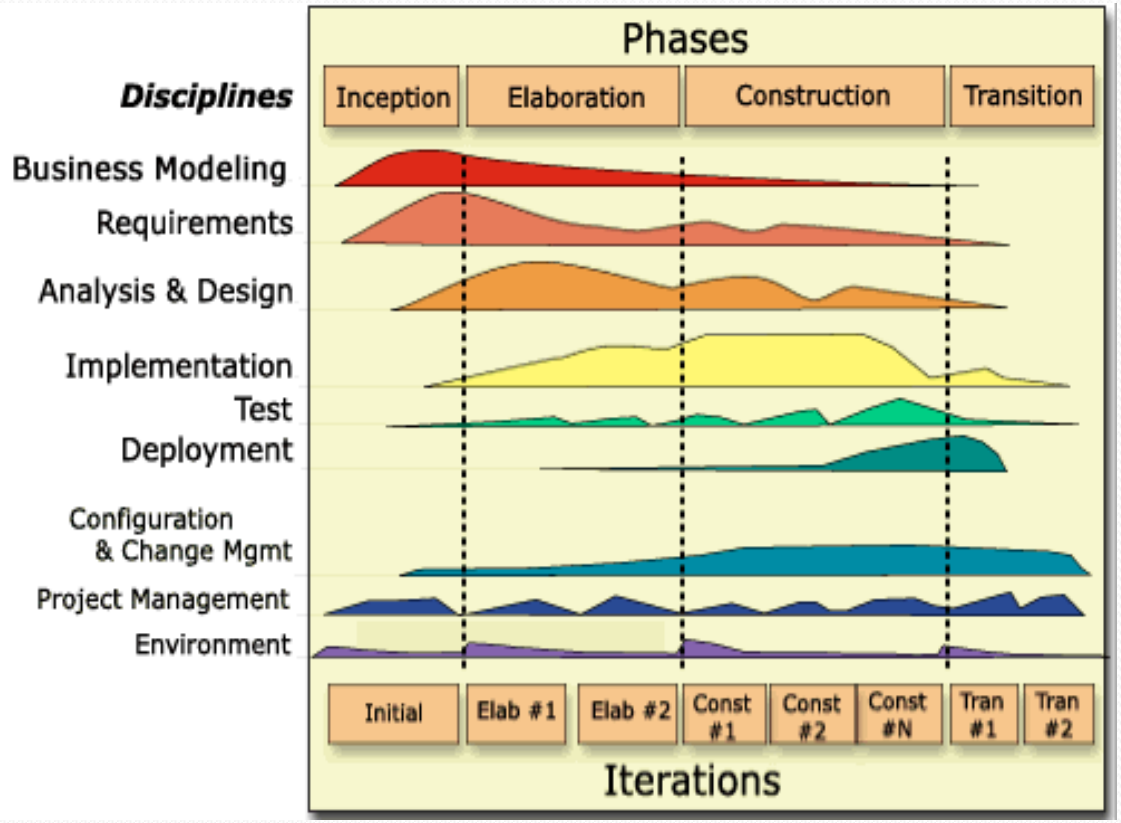
Rational Unified Process (RUP)

- **Rational Unified Process (RUP)** is a general process for developing object-oriented computer systems.
- It is a guide that shows how to practically use UML (Unified Modeling Language) to develop a computer system.
- RUP was created by **Rational** and now it is developed by **IMB**.
- The Rational Unified Process is a specific and detailed instance of a more generic process described by Ivar Jacobson, Grady Booch, and James Rumbaugh in the textbook, "***The Unified Software Development Process*** "

RUP phases

Horizontal axis: it represents **time** and it emphasizes the **dynamic aspects** of the process. On this axes the process is described in terms of cycles, phases, iterations and milestones.

Vertical axis: it represents the **static aspects of the process** and it is expressed in terms of: activities, products, workers and workflows.



1. Inception

Features:

- During the inception phase, you **establish the business case** for the system and delimit the **project scope**.
- To accomplish this you must identify all **external entities** with which the system will interact (**actors**) and define the nature of this interaction at a high-level.
- This involves **identifying all use cases** and **describing** a few significant ones.
- The business case includes **success criteria, risk assessment**, and estimate of the **resources** needed, and a **phase plan** showing dates of major milestones

Inception outcomes:

- A **VISION document**: a general vision of the core project's requirements, key features, and main constraints.
- An **initial use-case model** (10% -20%) complete).
- An **initial project glossary** (may optionally be partially expressed as a domain model).
- An **initial business case**, which includes business context, success criteria (revenue projection, market recognition, and so on), and financial forecast.
- An **initial risk assessment.**
- A **project plan**, showing phases and iterations.
- A **business model**, if necessary.
- **One or several prototypes**.

2. Elaboration phase

Features :

- The purpose of the elaboration phase is to analyze the problem domain, establish a sound **architectural foundation**, develop the **project plan**, and eliminate the highest risk elements of the project.
- The elaboration phase activities ensure that the **architecture, requirements and plans are stable enough**, and the risks are sufficiently mitigated, so you can predictably determine the cost and schedule for the completion of the development.
- In the elaboration phase, **an executable architecture prototype** is built in one or more iterations, depending on the scope, size, risk, and novelty of the project

Elaboration outcomes :

- **A use-case model** (at least 80% complete) — all use cases and actors have been identified, and most use case descriptions have been developed.
- **Supplementary requirements** capturing the non-functional requirements and any requirements that are not associated with a specific use case.
- A **Software Architecture Description**.
- An executable **architectural prototype**.
- A **revised risk list** and a **revised business case**.
- A **development plan** for the overall project, including the coarse-grained project plan, showing **iterations** and **evaluation criteria** for each iteration.
- An **updated development case** specifying the process to be used.
- A preliminary **user manual** (optional)

3. Construction phase

Features :

- All remaining components and application features are developed and integrated into the product, and all features are thoroughly tested.
- It is a manufacturing process where emphasis is placed on **managing resources** and **controlling operations** to optimize costs, schedules, and quality.
- The outcome of the construction phase is a product ready to put in hands of its end-users.

Construction outcomes:

- **The software product** integrated on the adequate platforms.
- The **user manuals**.
- A **description of the current release.**

- At this point, you decide if the software, the sites, and the users are ready to go operational, without exposing the project to high risks. This release is often called a "**beta**" **release**

4. Transition phase

Features :

- Typically, this phase includes several iterations, including **beta releases**, **general availability releases**, as well as **bug-fix** and enhancement releases.
- Considerable effort is expended in developing **user-oriented documentation**, **training users**, **supporting users** in their initial product use, and reacting to user feedback.
- This includes:
  - "**beta testing**" **to validate** the new system against user expectations
  - **parallel operation** with a legacy system that it is replacing
  - **conversion of operational databases**
  - **training** of users and maintainers
  - **roll-out the product** to the marketing, distribution, and sales teams.

Transition outcomes:

- **Installation plan**;
- **System final notes** ;
- **Documentation**.

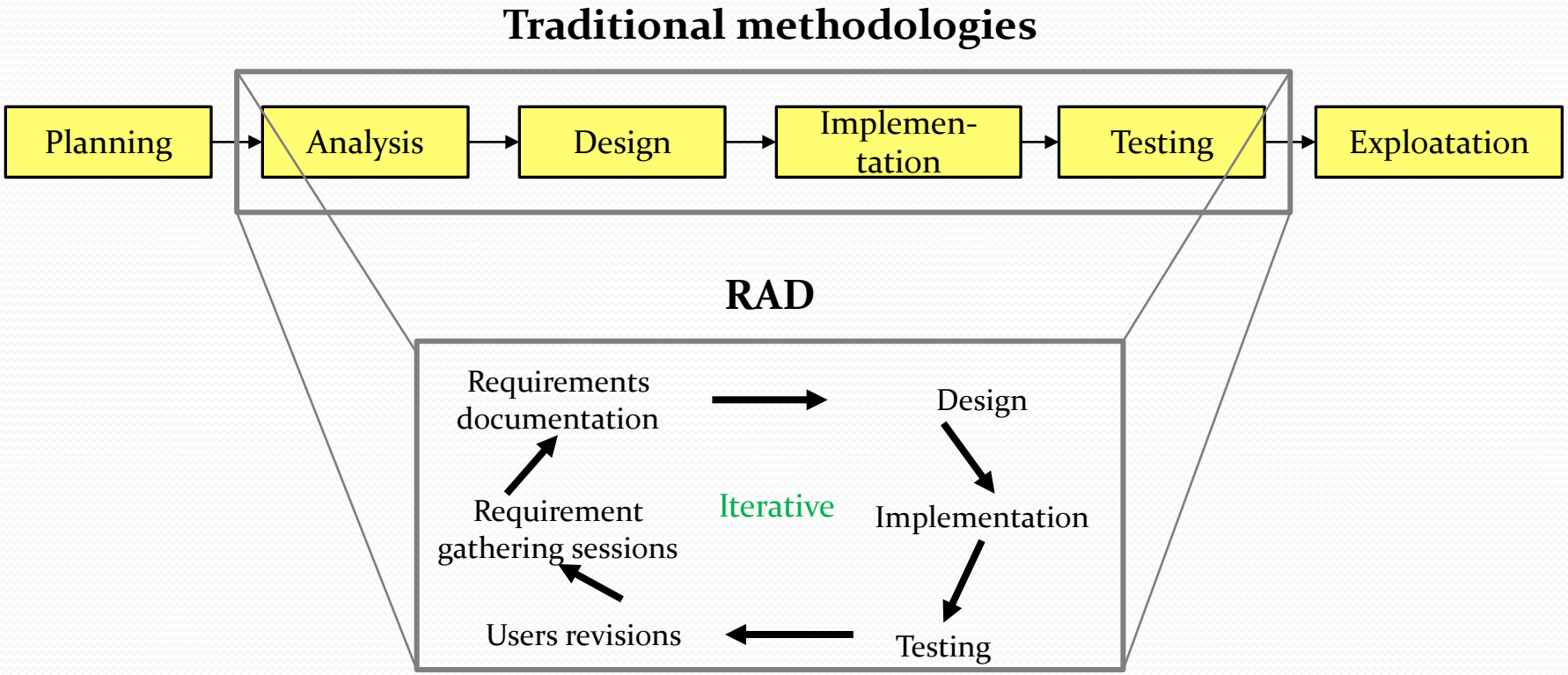
RAD (Rapid Application Development)

- RAD refers to a set of principles aimed at adjusting stages of development of computer systems so that part of the system to be developed and reach users quickly.
- Thus, users can better understand the system and suggest revisions that bring the system closer to its requirements.
- It is recommended that analysts use special techniques and tools to accelerate the stages of analysis, design and implementation, such as:
  - CASE tools
  - joint session to establish requirements - Join Requirement Planning (JRP)
  - fourth generation programming languages
  - visual programming languages
  - code generators
  - software component reuse

Today, almost all integrated development environments have RAD specific facilities.

RAD (Rapid Application Development)

- RAD compresses the steps of classic methodologies in an **iterative** process.
- It is based on **prototyping** and on user revisions before passing to a new iteration



Agile development methodologies

- These are **programming-oriented** methodologies and they have less rules and practices.
- All agile development methodologies are based on the agile manifesto and a set if twelve principles:

1. Our highest priority is to satisfy the **customer** through early and continuous delivery of valuable software.
2. Welcome **changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver** working software **frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must **work together daily** throughout the project.
5. Build projects around **motivated individuals**. Give them the environment and support they need,and trust them to get the job done.

1. http://agilemanifesto.org/principles.html

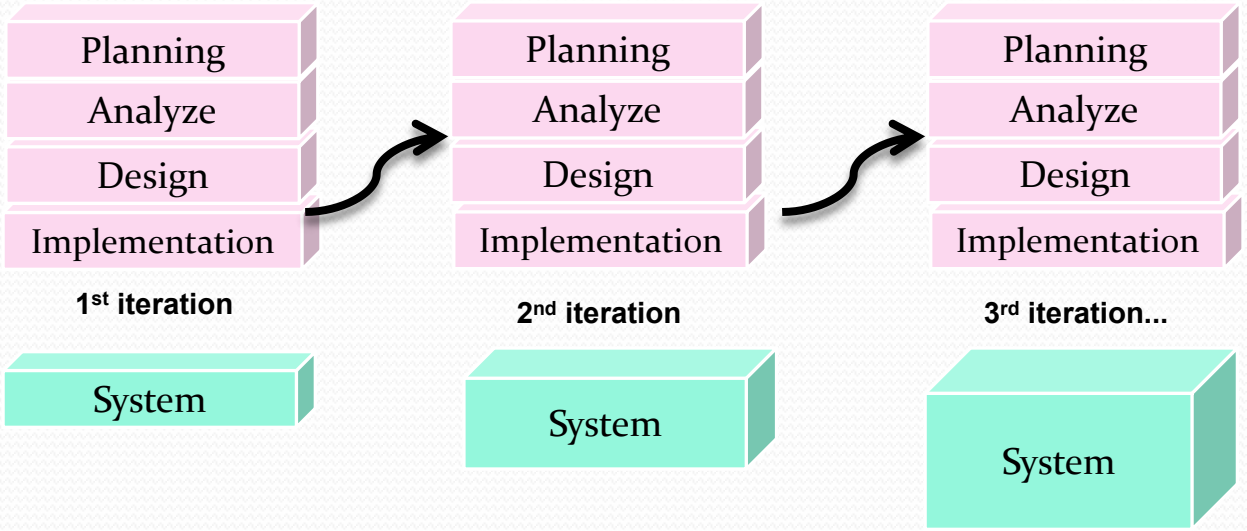
Agile development methodologies

Principles of the agile manifesto (cont.):

6. The most efficient and effective method of conveying information to and within a development team is **face-to-face** conversation.
7. **Working software** is the primary measure of progress.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical **excellence and good design** enhances agility.
10. **Simplicity** -- the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing teams.**
12. At regular intervals, the team **reflects** on how to become more effective, then tunes and **adjusts** its behavior accordingly.

Agile development methodologies

- Starting from those principles, agile methodologies are focused on system development optimization by removing an important part of the modelling and documentation activities.
- It is supported the simple, iterative development of systems. Virtually, all the agile methodologies are used in combination with object-oriented techniques.
- All agile development methodologies have a simple development cycle, including the traditional stages of the system development process.
- The most popular examples of agile methodologies are Extreme Programming (XP) and Scrum.

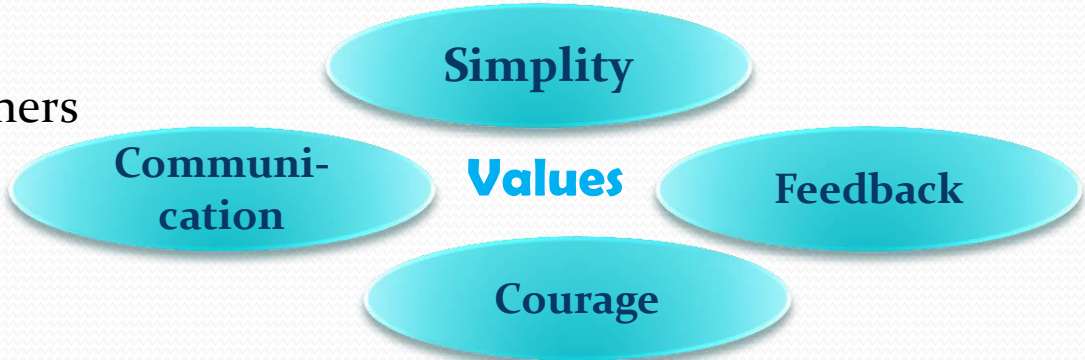


Agile development methodologies

Advantages	Disadvantages
Realistic approach in computer system development.	They are not appropriate to manage complex dependencies.
It promotes team work and learning..	Increased risk of sustainability, maintainability and extensibility.
Functionalities can be quickly implemented and checked.	Poor documentation means that neither the system, neither the development process cannot be audited.
It is appropriate model for continuously changing environments.	They depend heavily on interaction with the customer.
Minimum rules, easy to achieve documentation	Information transfer to new team members is hampered by lack of documentation.
Easy to manage.	Lack of rules can result in a chaotic work environment.
Great flexibility.	The team depends of the team members who know the system requirements.

Extreme Programming - XP

- It emphasized coding (standards, principles) using a common set of names, descriptions and coding practices.
- Claims that programmers work in pairs ("pair programming") and programmers have share a responsibility for each software component developed.
- Many discussion sessions during development.
- Continuous rapid feedback of end users continuous.
- Developers must have a quality-oriented mentality.
- It relies heavily on refactoring, which is a disciplined way to restructure the code to keep it simple.
- The system is developed in a manner evolutionary and incremental.
- Each iteration (1-4 weeks) has a functional outcome.
- Low support for modelling.
- Close relationship between customers and developers.
- Lack of documentation of system development.



Extreme Programming - XP

It is appropriate for...

- **Small projects** with **highly motivated, united, stable, and experienced team**
- **Small groups of developers**, encountering less that 10 persons
- For **short development cycles**, when **frequent discussions with final users** are facilitated

It is not appropriate...

- In the case the project is not small or the **teams are not united**, the development success is doubtful.
- There are doubts about the benefits of introducing **external contractors** within an existing team when working under XP.
- XP requires a **high degree of discipline**; otherwise projects will become unfocused and chaotic
- Not recommended for **large applications**. Lack of analysis and design documentation (there exists only code documentation associated with XP) may lead to the impossibility of assuring maintenance to a big system..

SCRUM

- The methodology name is taken from Rugby and designates a scrum (an ordered group of players) used to restart a game
- SCRUM creators believe that no matter how well a system development planning is carried out, as soon as the software is beginning to be developed the chaos will break out and plans will not have utility.

Organizing principles

- Teams are self-organized and self-directed.
- Unlike other approaches, Scrum teams do not have a designated team leader.
- Teams are organized in a symbiotic manner and set **their own goals** for each **sprint** (iteration).

SCRUM

Functioning principles

- Once a sprint started, Scrum teams **do not take into account any additional requirement.**
- Any new requirements that are discovered are placed in a **list of requirements to be addressed.**
- At the beginning of each working day, there is a meeting where all team members **sit in a circle** and report **achievements** of the previous day, determine what they're going to do today and describe everything that has blocked progress during the previous day.
- To ensure continuous progress, any identified blockage is addresses in the next hour.
- At the end of each sprint, the team presents the software to the client.
- Based on the results of the finished iteration, a new plan is conceived for the new iteration.