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
- Object-oriented methodologies
  - Iterative development of computer systems using UML
  - Unified methodology for computer system development (RUP)
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**
## Concepts used in computer system development

<table>
<thead>
<tr>
<th>Concept</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process/stage</td>
<td>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.</td>
</tr>
<tr>
<td>Activity</td>
<td>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</td>
</tr>
<tr>
<td>Phase</td>
<td>It represents the time between two key points of a process, while a well-defined set of objectives is achieved in Rational Unified Process.</td>
</tr>
<tr>
<td>Step</td>
<td>It represents a sequence of activities carried on in a work stage – in SSADM</td>
</tr>
</tbody>
</table>
### Concepte utilizate în realizarea sistemelor informatice

<table>
<thead>
<tr>
<th>Concept</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tasks</strong></td>
<td>They are <em>components of activities</em> and are a set of actions that constitutes <em>the responsibility of a persons</em> or <em>group of persons</em>. A task has preconditions, deliverables and postconditions. The succession in time of tasks, activities, stages and processes form the <em>lifecycle of the computer system</em>.</td>
</tr>
<tr>
<td><strong>Lifecycle of a computer system</strong></td>
<td>It is defined by the succession in time of tasks, activities, stages and processes. It is a <em>template for ordering the activities</em> for computer system development, covering the <em>time interval</em> 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.</td>
</tr>
<tr>
<td><strong>Development cycle of a computer system</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
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 Réalisation 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 analyst 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 to the resumption of the whole study.
Structured approach methodologies- examples

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 Réalisation, Informatique pour les Systèmes d’Entreprise MERISE.**
- **Jackson System Development (JSD).**
- **Information System Work and Analysis of Changes (ISAC).**
- **Effective Technical 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 be 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.
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 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 off from the design phase of the system.
IV. Evolutive lifecycle model

- It is recommended for complex systems that can be decomposed in subsystems. 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 applies 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 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

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 module</td>
<td>Feasibility study</td>
</tr>
<tr>
<td>1st module</td>
<td>Requirements analysis</td>
</tr>
<tr>
<td>2nd module</td>
<td>Requirements specification</td>
</tr>
<tr>
<td>3rd module</td>
<td>Logical system specification</td>
</tr>
<tr>
<td>4th module</td>
<td>Physical design</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 stage</td>
<td>Feasibility study</td>
</tr>
<tr>
<td>1st stage</td>
<td>Current environment investigation</td>
</tr>
<tr>
<td>2nd stage</td>
<td>Business system options</td>
</tr>
<tr>
<td>3rd stage</td>
<td>Requirements definition</td>
</tr>
<tr>
<td>4th stage</td>
<td>System technical options</td>
</tr>
<tr>
<td>5th stage</td>
<td>Logical design</td>
</tr>
<tr>
<td>6th stage</td>
<td>Physical design</td>
</tr>
</tbody>
</table>
### Exemplle of iterative process for computer system development with UML (Unified Modeling Language)

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Problem identification</strong></td>
<td>It identifies the main characteristics of the studied economic unit and the mode of operation of the implemented activity.</td>
</tr>
<tr>
<td><strong>Solution structuring</strong></td>
<td>Beneficiary requirements are detailed. It includes the following steps: <strong>Actor setting; Use case setting; setting relationships between use cases; Use case diagrams construction.</strong></td>
</tr>
<tr>
<td><strong>System analysis</strong></td>
<td>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: <strong>class diagram, object diagrams, state chart diagrams, activity diagrams, sequence diagrams, communication diagrams.</strong></td>
</tr>
<tr>
<td><strong>System design</strong></td>
<td>It has two sub-stages: <strong>architecture design</strong> and <strong>detailed design.</strong> It involves system architecture design, database design, interface design, algorithm design. The following diagrams are built: <strong>component diagrams and deployment diagrams.</strong></td>
</tr>
<tr>
<td><strong>System implementation</strong></td>
<td>It involves the actual programming of the identified classes by writing <strong>source code</strong> and implementing <strong>forms and reports.</strong></td>
</tr>
</tbody>
</table>
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:

- Establish the **business case** for the system and delimit the **project scope**.
- 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**, an **initial use-case model** (10% -20%) complete).
- An **initial project glossary**
- 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:

- 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)
- 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.
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 the focus is put 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** users and maintainers
  - **roll-out the product** to the marketing, distribution, and sales teams.

Transition outcomes:
- **Installation plan**;
- **System final notes**;
- **Documentation**.
Modeling in computer system development

- Modeling is one of the most important computer system design techniques.
- A *model* represents an abstractization of a part of the reality in order to better understand the software product that must be developed before starting the actual construction.
- Once built, the model serves for communication within the development team and externally, with users.
- Characteristic features of modeling are: simplification, subordination to a purpose, representation of a reality, division, hierarchy and communication.
Modeling in object-oriented system development

- Object oriented analysis and design uses **three types of models** for describing computer system:
  - *Static model* describes system objects and their relationships;
  - *Dynamic model* describes object interactions within the system;
  - *Functional model* describes data value transformation within the system.
Languages for information modeling

- Generally speaking, languages for information modeling enable **concise and accurate description** of system properties at various level of abstractization.
- They are an integrated part of the computer system development process and they can be used for:
  - linking the **analysis** phase with the **requirements specification** and **implementation** phases;
  - checking the critical properties of systems;
  - assisting in **automatrical generation of code and test cases**.
Categories of languages for information modeling

Depending on the required formalization level, the languages for information modeling are divided in:

- **Informal languages**: natural language
- **Semi-formal languages**: UML, BPMN or SysML
- **Formal languages**: OCL (Object Constraint Language), Z

<table>
<thead>
<tr>
<th>Syntax rules</th>
<th>NO</th>
<th>YES</th>
<th>YES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic rules</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Requirement specification language</td>
<td>Informal (natural language)</td>
<td>Semi-formal (UML)</td>
<td>Formal (Z, OCL)</td>
</tr>
</tbody>
</table>
UML language

- **UML (Unified Modeling Language)** It arose from the need to standardize the typology, semantics and representation of results.

- Currently, UML is a modeling standard recognized by OMG (Object Management Group). Standardization is conducted from November 1997, a continuous improvement being performed so far.

- UML can be defined as a visualization, specification, model building and documentation language. Its main value comes from being an open standard, covering the whole software development cycle and many application types. It is based on the experience of the team that developed it and can be implemented by many CASE tools.
UML language

- UML has **standard notations** and **semantic** that are appropriate for object-oriented system modeling.

- With the advent of this language, system designers can easier understand the **system documentation**.

- Before this standardization, an object-oriented project could be described using one of the multitude of available object-oriented methods. If a **revision** was needed, a lot of time would have been lost with analyzing **method notation** and **semantic**, before starting the design.
UML history

- **1990**: Ada/Booch, Booch '91, Booch '93
- **1995**: UML 0.8, UML 0.9, OMT '94, OOSE '94
- **1997**
  - **1997**: UML 1.1, Unified Process
  - **2000**: UML 1.3, UML 1.4
  - **2003**: UML 1.5
  - **2005**: UML 2.0
- **2008**: UML 2.1.2, UML 2.2, SysML 1.1, BPMN 1.1, DSLs
- **2011**: UML 2.4.1
UML – basic elements

1. Metamodel for object oriented modeling
   - A coherent set of **concepts** and **relationships** between them;
   - Each element used for modeling is defined using a **precise syntax** (E.g.: a class definition);
   - It is a support language for **transmitting visual models** between various tools;
UML basic elements

2. Diagram types
3. Extension mechanisms

- **Stereotypes** are attached to an element of the model or a relation between elements (there are predefined stereotypes).

- **Notes** offer a supplementary description of a model element.

- **Constraints** limit the use of a model element.

- **Tagged values** represent attributes that are defined for a specific stereotype.

- **Profiles** personalize the metamodel by constructions that are specific to a particular field of activity, platform or development method.
Modeling with UML diagrams - perspectives

A. *Business process modeling* is performed by *use case diagram*. This diagram conducts the whole development process in the case of *use case oriented methods*.

B. *Static structure modeling* is performed using *class diagram* (for modeling the static structure of system classes) and *object diagrams* (for modeling the static structure of system objects). *Package diagram* is a mean of grouping diagram elements in packages.
C. *Dynamic modeling* is performed by:

i. interaction diagrams
   - SEQUENCE DIAGRAM for modeling message flows between objects and
   - COMMUNICATION DIAGRAM for modeling object interactions.

ii. diagrams that describe system behavior -
   - STATECHART DIAGRAM for modeling system object behavior and
   - ACTIVITY DIAGRAM for modeling the behavior of uses cases, objects or operations;

D. *Implementation modeling* is performed by two diagrams:

- COMPONENT DIAGRAM – for modeling software components
- DEPLOYMENT DIAGRAM for modeling system physical distribution
CASE tools are software packages based on a certain methodology/method that support designers in developing a software product.

CASE tools usefulness is:

- **quantitative** (reducing time and cost of implementation) and
- **qualitative** (rigorous application of a development methodology, reducing shortfalls in implementation, achieving standard documentation, ease of redesign).

CASE tools substantially reduce or eliminate many of the design and development problems of computer applications.
# Traditional development vs. CASE based development

<table>
<thead>
<tr>
<th>Computer system traditional development</th>
<th>CASE-based computer system development</th>
</tr>
</thead>
<tbody>
<tr>
<td>It focuses on coding and testing</td>
<td>It focuses on analysis and design.</td>
</tr>
<tr>
<td>The specifications are based on paper.</td>
<td>Quick interactive prototyping</td>
</tr>
<tr>
<td>Program manual coding</td>
<td>Automatic code generation</td>
</tr>
<tr>
<td>Manual documentation generation</td>
<td>Automatic documentation generation</td>
</tr>
<tr>
<td>Continuous testing of the software</td>
<td>Automatic validation</td>
</tr>
<tr>
<td>Maintenance of code and documentation</td>
<td>Maintenance of design specifications</td>
</tr>
</tbody>
</table>