# SYSTEMS DEVELOPMENT METHODOLOGIES

There are many ways to **categorize methodologies**. One way is by looking at whether they **focus** on business **processes or the data** that support the business. Another important factor in categorizing methodologies is the **sequencing of the SDLC phases** and the **amount of time and effort** devoted to each.

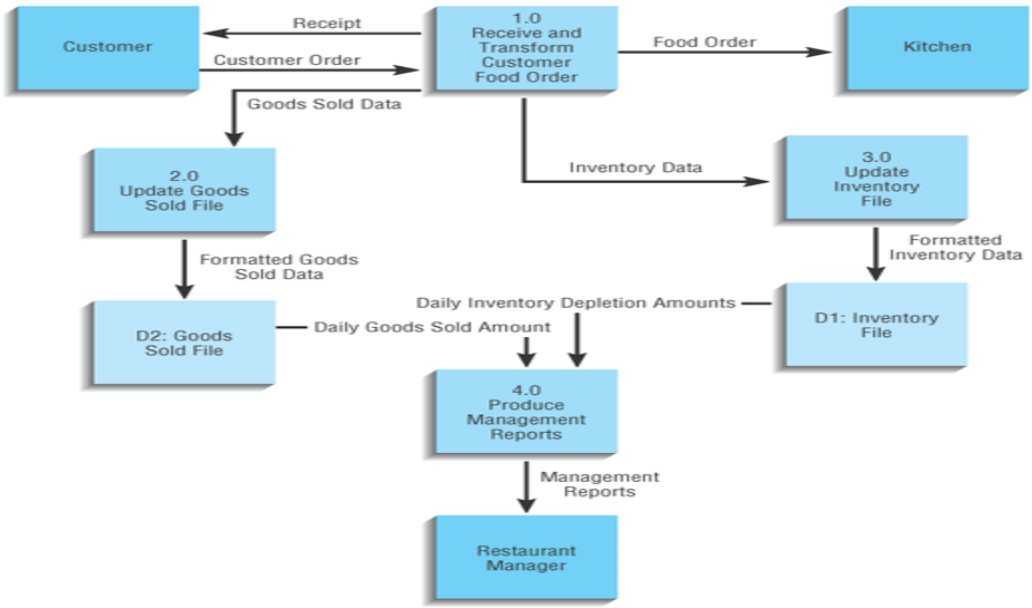
## Structured analysis and design methodology

The structured analysis and design analysis and design methodology is essentially a **process modeling based-SDLC** methodology. **Process modeling involves graphically representing the processes or actions**, that capture, manipulate, store and distribute data between the system and its environment and among components within the system.

This technique / methodology describes the system through a set of process models therefore; it is also known **as *process-centric*** method. System is viewed from perspective of data flowing through different processes. Function of a system is described by processes that transform data into useful information which leads to achieve the systems objectives.

A common form of process model is a data flow diagram (DFD). Data flow diagram is a graphic that illustrates the movement of data between external entities and the processes and the data stores within a system.

**Figure 1.10** A Data Flow Diagram



The **SDLC considers** systems development as a ***single cycle*** of sequential steps. (planning and selection, analysis, design, implementation and operation)

The structured approach is also based on **decomposition** concept. The system is decomposed into sub processes, sub processes into further sub processes and so on. This allows concentrating on the relevant part only by keeping other details aside.

DFDs were the common mechanisms for systems analysis during the 1980s.They are excellent representational tools for studying processes and systems analysts should not ignore them simply because they are not the latest **fad** in the market.

Though successful, it has **limitations** in that it is tuned for process modeling. However, Processes are not as stable as data and changes in processes can lead to substantial maintenance.

> **NB; Since it has minimal focus on data modelling it does not lend itself easily to DBMS based systems**

**Data Oriented Methodology**

Another popular approach during the 1980s used principles similar to structured analysis and design but provided more emphasis on data.

Its **data centered but process sensitive methodology** that tries to minimize redundancy of data in an organization.

Emphasis **data models** as the core of the concept. Depicts *ideal organization of data*, independent of where and how data are used. Data model describes kinds of **data** and business **relationships among the data**. Business rules depict how organization captures and processes the data.

Models related with this methodology includes the **E-R model** and the **relational model.** (popular with relational-model based DBMS). This methodology prescribed modeling of data before processes on the premise that data is inherently more stable than processes

**Object Oriented Methodology**

An analysis and design approach based on the **notion of an object**, which captures the data and processes of a thing in a single construct.

In traditional approaches, the problem-decomposition process is either process-centric or data-centric i.e. data and processes are separated. Analysts found this division unnatural because real world objects have **properties** (akin to data) and **behavior** (akin to processes). Using this perspective there

was no reason to separate the two in information systems world. **The separation mitigated reuse**. Processes and data are so closely related that it is difficult to pick one or the other as the primary focus.

Based on this lack of congruence with the real world and in an attempt to facilitate **reuse and improve development productivity**, object-oriented analysis and design approach (OOSAD) was proposed. OOSAD attempt to balance the emphasis between process and data by focusing the decomposition of problems on **objects** that contain both data and processes.

The growing popularity of OOSAD coincided with the emergence of Unified modeling language (UML).

Unified modeling language is a language that has been adopted by **object management group (OMG)** which includes some of the largest software vendors in the World i.e. Apple corp., HP, data general, American airlines, IBM etc. **OMG was founded in 1989**. UML unifies features from some of the leading proponents of OOSAD especially those of Grady Booch, Ivar Jacobson, and James Rumbaugh who are fondly known as **the three amigos** of the object modeling approach

***Definition of UML***
*A notation that allows the modeler to **specify, visualize, and construct** the artifacts of software systems, as well as business models.*

**More on Object-Oriented Analysis and Design Approach (OOSAD)**

According to the creators of the Unified Modeling Language (UML), any modern object-oriented approach to developing information systems must be **use-case driven, architecture-centric, and iterative and incremental.**

**Use-Case Driven**
Use-case driven means that **use cases** are the **primary modeling tools defining the behavior of the system**. A use case describes how the user interacts with the system to perform some activity, such as placing an order, making a reservation, or searching for information.

The use cases are used to identify and to communicate the requirements for the system to the programmers who must write the system. ***Use cases are inherently simple because they focus on only one business process at a time***. In contrast, the process model diagrams used by traditional structured and RAD methodologies are far more complex because they require the systems analyst and user to develop models of the entire system.

With traditional methodologies, each system is decomposed into a set of subsystems, which are, in turn, decomposed into further subsystems, and so on. This goes on until no further process decomposition makes sense, and it often requires dozens of pages of interlocking diagrams. In contrast, a use case focuses on only one business process at a time, so developing models is much simpler.

**Architecture-centric**

Any modern approach to systems analysis and design should be architecture-centric. Architecture-centric means that the underlying software architecture of the evolving system specification drives the specification, construction, and documentation of the system.

Modern object-oriented systems analysis and design approaches should support **at least three separate but interrelated architectural views of a system: functional, static, and dynamic**.

*NB; The functional, or external, view describes the behavior of the system from the perspective of the user. The structural, or static, view describes the system in terms of attributes, methods, classes, and relationships. The behavioral, or dynamic, view describes the behavior of the system in terms of messages passed among objects and state changes within an object.*

**Iterative and Incremental**

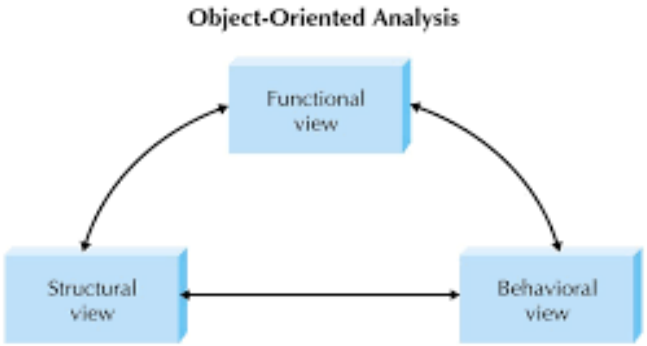
Modern object-oriented systems analysis and design approaches emphasize iterative and incremental development that undergoes continuous testing and refinement throughout the life of the project.

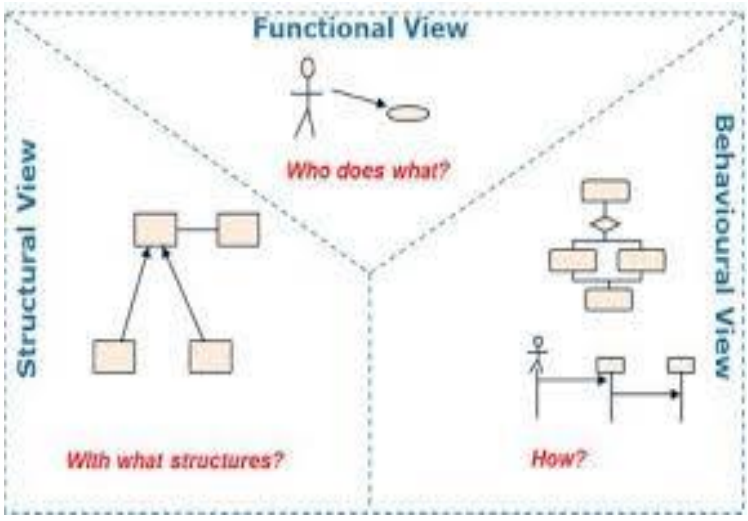
This implies that the systems analysts develop their understanding of a user's problem by **building up the three architectural views little by little**. The systems analyst does this by working with the user to create a functional representation of the system under study. Next, the analyst attempts to build a structural representation of the evolving system. Using the structural representation of the system, the analyst distributes the functionality of the system over the evolving structure to create a behavioral representation of the evolving system.

As an analyst works with the user in developing the three architectural views of the evolving system, the analyst iterates over each of and among the views. That is, as the analyst better understands the structural and behavioral views, the analyst uncovers missing requirements or misrepresentations in the

functional view. This, in turn, can cause changes to be cascaded back through the structural and behavioral views.

All three architectural views of the system are interlinked and dependent on each other (see Figures below). As each increment and iteration is completed, a more-complete representation of the user's real functional requirements is uncovered.




## Benefits of Object-Oriented Systems Analysis and Design

Concepts in the object-oriented approach enable analysts to break a complex system into smaller, more-manageable modules, work on the modules individually, and easily piece the modules back together to form an information system.

This **modularity** makes systems development **easier to grasp, easier to share among members of a project team**, and easier to **communicate to users**, who are needed to provide requirements and confirm how well the system meets the requirements throughout the systems development process.

By modularizing systems development, **the project team actually is creating reusable pieces** that can be plugged into other systems efforts or used as starting points for other projects.

As the programmer has to spend less time and effort so he can utilize saved time (due to the reusability feature of the OOM) in concentrating on other aspects of the system.

**Table 1.2** Key Differences Between Structured Analysis and Design and Object-Oriented Analysis and Design

| Characteristics | Structured Analysis and Design | Object-Oriented Systems Analysis and Design |
|---|---|---|
| Methodology | SDLC | Iterative/Incremental |
| Focus | Processes | Objects |
| Risk | High | Low |
| Reuse | Low | High |
| Maturity | Mature and widespread | Emerging |
| Suitable for: | Well defined projects with stable user requirements | Risky large projects with changing user requirements |

## THE RATIONAL UNIFIED PROCESS (RUP)

The Unified Process is a specific methodology that maps out when and how to use the various Unified Modeling Language (UML) techniques for object-oriented analysis and design.

Whereas the UML provides structural support for developing the structure and behavior of an information system, the Unified Process provides the behavioral support. The Unified Process, of course, is use-case driven, architecture-centric, **and iterative and incremental**. Furthermore, the Unified Process is

a two-dimensional systems development process described by a set of phases and workflows.

The phases are **inception, elaboration, construction, and transition**. The workflows include business modeling, requirements, analysis, design, implementation, test, deployment, configuration and change management, project management, and environment. Figure 1-15 depicts the Unified Process.

**Iterative and Incremental development**

A key feature of RUP is the iterative and incremental nature of the development process.

-**Iterative** – means developing in pieces, and growing and *improving each piece* through successive iterations.

**Incremental** – means developing in pieces, and progressively *adding new pieces* through successive iterations.

**Phases**

The phases of the Unified Process support an analyst in developing information systems in an iterative and incremental manner. The phases describe how an information system evolves through time. Depending on which development phase the evolving system is currently in, the level of activity varies over the workflows.

The curve in Figure 1-15 **associated with each workflow approximates the amount of activity that takes place during the specific phase. For example, the inception phase primarily involves the business modeling and requirements workflows, while practically ignoring the test and deployment workflows.** Each phase contains a set of iterations, and each iteration uses the various workflows to create an incremental version of the evolving system. As the system evolves through the phases, it improves and becomes more complete. Each phase has objectives, a focus of activity over the workflows, and incremental deliverables. Each of the phases is described next

**Inception**

In many ways, the inception phase is very similar to the planning phase of a traditional SDLC approach. In this phase, a business case is made for the proposed system. This includes feasibility analysis that should answer questions such as the following:

Do we have the technical capability to build it (technical feasibility)?
If we build it, will it provide business value (economic feasibility)?
If we build it, will it be used by the organization (organizational feasibility)?

To answer these questions, the development team performs work related primarily to the business modeling, requirements, and analysis workflows. In some cases, depending on the technical difficulties that could be encountered during the development of the system, a throwaway **prototype** is developed.

This implies that the design, implementation, and test workflows could also be involved. The project management and environment supporting workflows are very relevant to this phase. The primary deliverables from the inception phase are **a vision document** that sets the scope of the project; identifies the primary requirements and constraints; sets up an initial project plan; and describes the feasibility of and risks associated with the project, the adoption of the necessary environment to develop the system, and some aspects of the problem domain classes being implemented and tested.

Summary:
-Defining the scope, determining the feasibility, understanding user requirements, preparing a software development plan
-Relatively short, low resource requirements
-Focus on planning and initial analysis

**Elaboration**

When we typically think about object-oriented systems analysis and design, the activities related to the elaboration phase of the Unified Process are the most relevant. The **analysis and design workflows** are the primary focus during this phase. The elaboration phase continues with developing the vision document, including finalizing the business case, revising the risk assessment, and completing a project plan in sufficient detail to allow the stakeholders to be able to agree with constructing the actual final system.

It deals with gathering the requirements, building the UML structural and behavioral models of the problem domain, and detailing how the problem domain models fit into the evolving system architecture. Developers are involved with all but the deployment engineering workflow in this phase. As the developers iterate over the workflows, the importance of addressing configuration and change management becomes apparent.

Also, the development tools acquired during the inception phase become critical to the success of the project during this phase. The primary deliverables of this phase include the **UML structure and behavior diagrams and an executable of a baseline version** of the evolving information system. The baseline version serves as the foundation for all later iterations. By providing a solid foundation at this point, the developers have a basis for completing the system in the construction and transition phases.

Summary:
-Detailed user requirements and baseline architecture is established
-Fairly long, but not high in resource demand
-Focus on detail analysis and design

**Construction**

The construction phase focuses heavily on programming the evolving information system. This phase is primarily concerned with the implementation workflow. However, the requirements workflow and the analysis and design workflows also are involved with this phase. It is during this phase that missing requirements are identified and the analysis and design models are finally completed.
Typically, there are iterations of the workflows during this phase, and during the last iteration, the deployment workflow kicks into high gear. The configuration and change management workflow, with its version-control activities, becomes extremely important during the construction phase. At times, an iteration has to be rolled back. Without good version controls, rolling back to a previous version (incremental implementation) of the system is nearly impossible. The primary deliverable of this phase is an **implementation** of the system that can be **released for beta and acceptance testing**

Summary:
-Coding, testing, and documenting code
-Longest and most resource-intensive
-Focus is on design and implementation

**Transition**

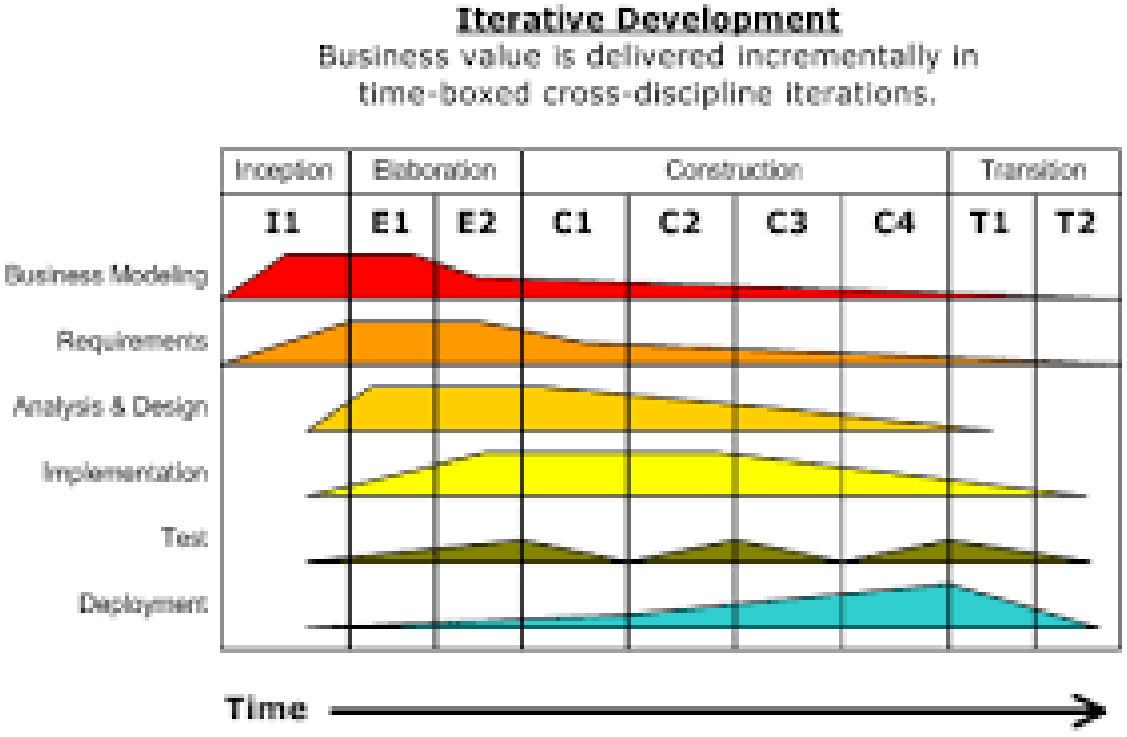
Like the construction phase, the transition phase addresses aspects typically associated with the implementation phase of a traditional SDLC approach. Its primary focus is on the testing and deployment workflows. Essentially, the business modeling, requirements, and analysis workflows should have been completed in earlier iterations of the evolving information system.

Furthermore, the testing workflow will have been executing during the earlier phases of the evolving system.

Depending on the results from the testing workflow, some redesign and programming activities on the design and implementation workflows could be necessary, but they should be minimal at this point. From a managerial perspective, the project management, configuration and change management, and environment are involved. Some of the activities that take place are beta and acceptance testing, fi ne-tuning the design and implementation, user training, and rolling out the final product onto a production platform.

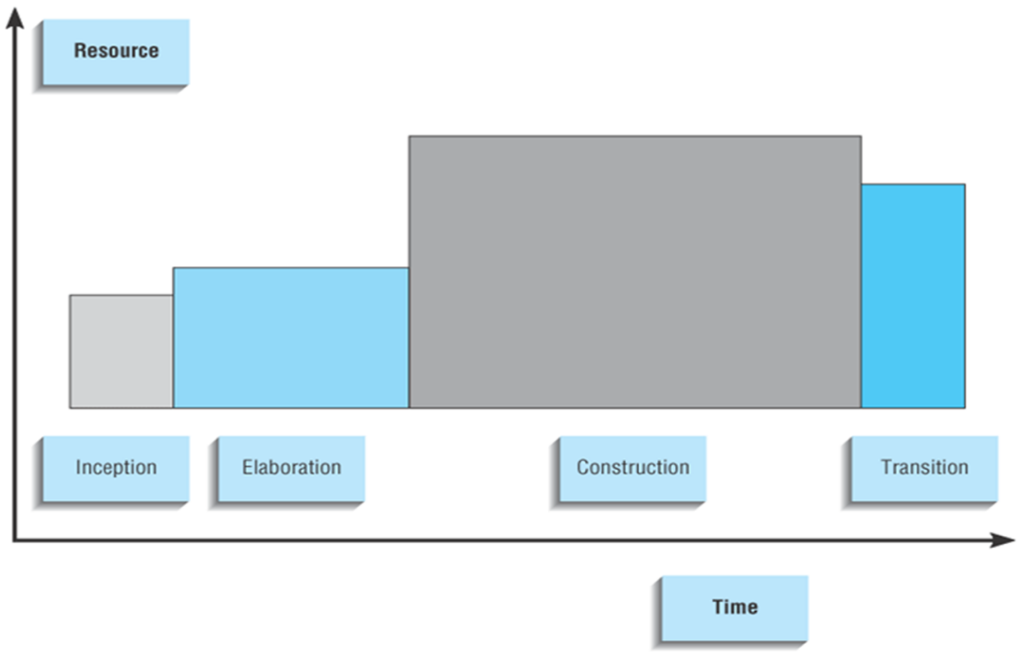
Obviously, the primary deliverable is the **actual executable information system**. The other deliverables include **user manuals**, **a plan to support the users, and a plan for upgrading the information system in the future.**

*Summary:*
-System is deployed and users are trained and supported
-Short-term, but resource-intensive
-Focus is on installation, training, and support



**NB; based on time and resource usage the four phases can graphically be represented as follows**

**Figure 1.12** Phases of OOSAD-Based Development




## Workflows

The workflows describe the tasks or activities that a developer performs to evolve an information system over time. The workflows of the Unified Process are grouped into two broad categories: engineering and supporting.

The core engineering and support workflows are the followings:

1. Business modelling. The business processes are modelled using business use cases.
2. Requirements. Actors who interact with the system are identified and use cases are developed to model the system requirements.

3. Analysis and design. A design model is created and documented using architectural models, component models, object models and sequence models.
4. Implementation. The components in the system are implemented and structured into implementation sub-systems. Automatic code generation from design models helps accelerate this process.
5. Testing. Testing is an iterative process that is carried out in conjunction with implementation. System testing follows the completion of the implementation.
6. Deployment. A product release is created, distributed to users and installed in their workplace.
7. Configuration and change management. This supporting workflow manages changes to the system.
8. Project management. This supporting workflow manages the system development.
9. Environment. This workflow is concerned with making appropriate software tools available to the software development team.