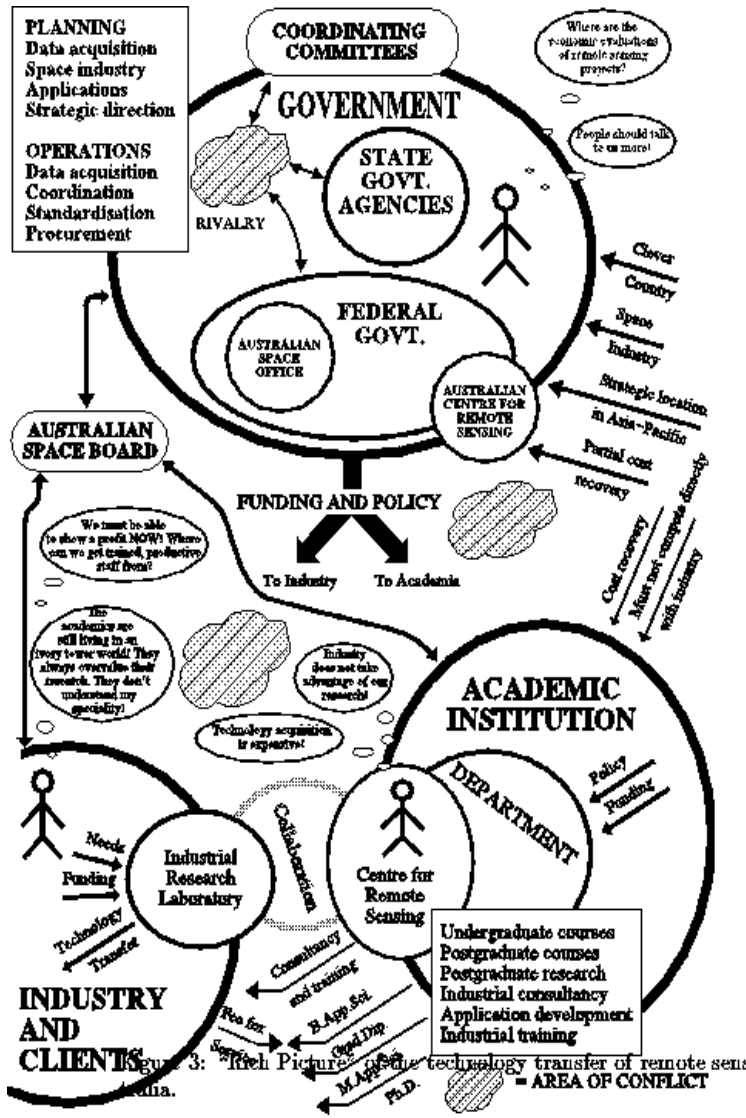


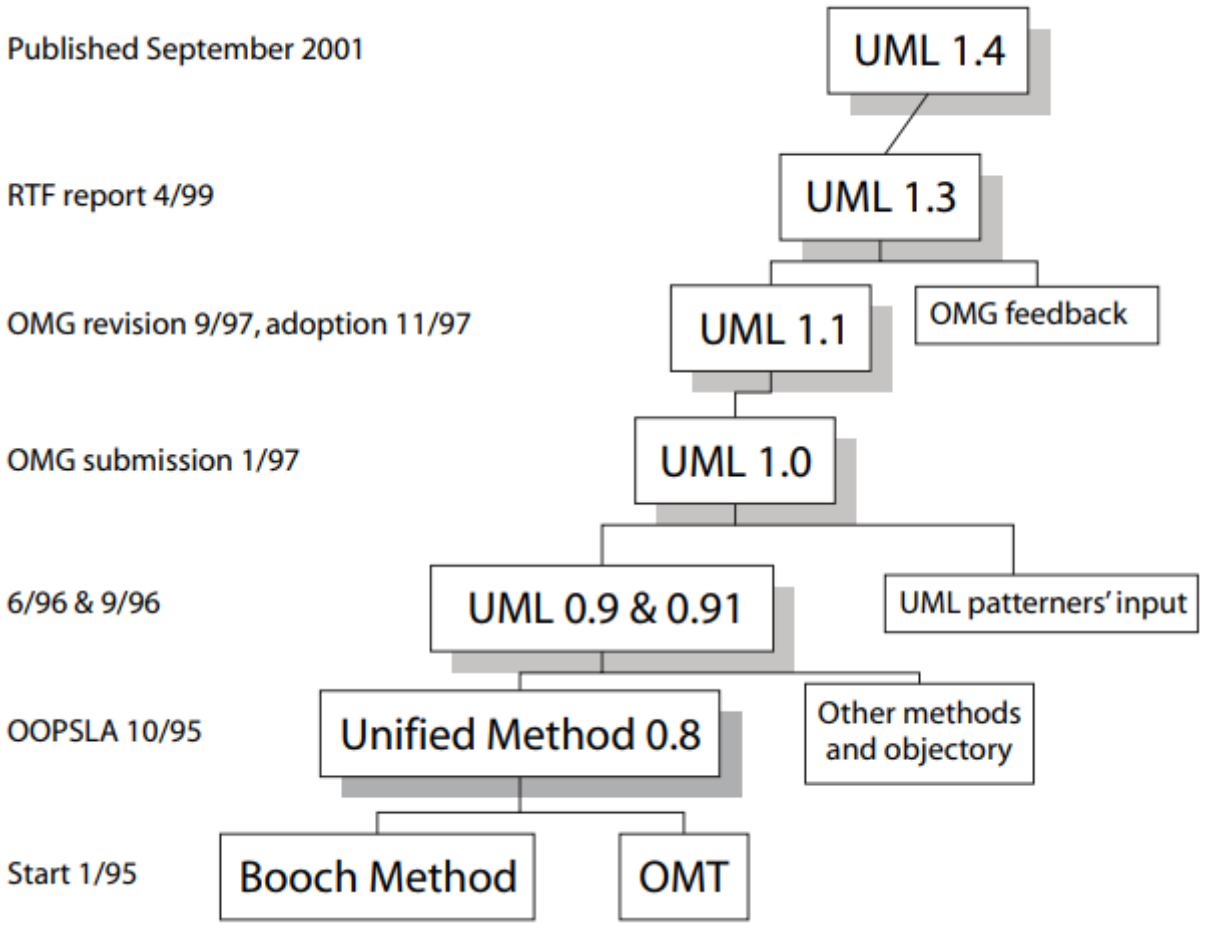
Figure 3: Rich Picture of the technology transfer of remote sensing in Australia.

Software Development

Process, Models,
Methods,
Diagrams
Software Development
Life Cyles

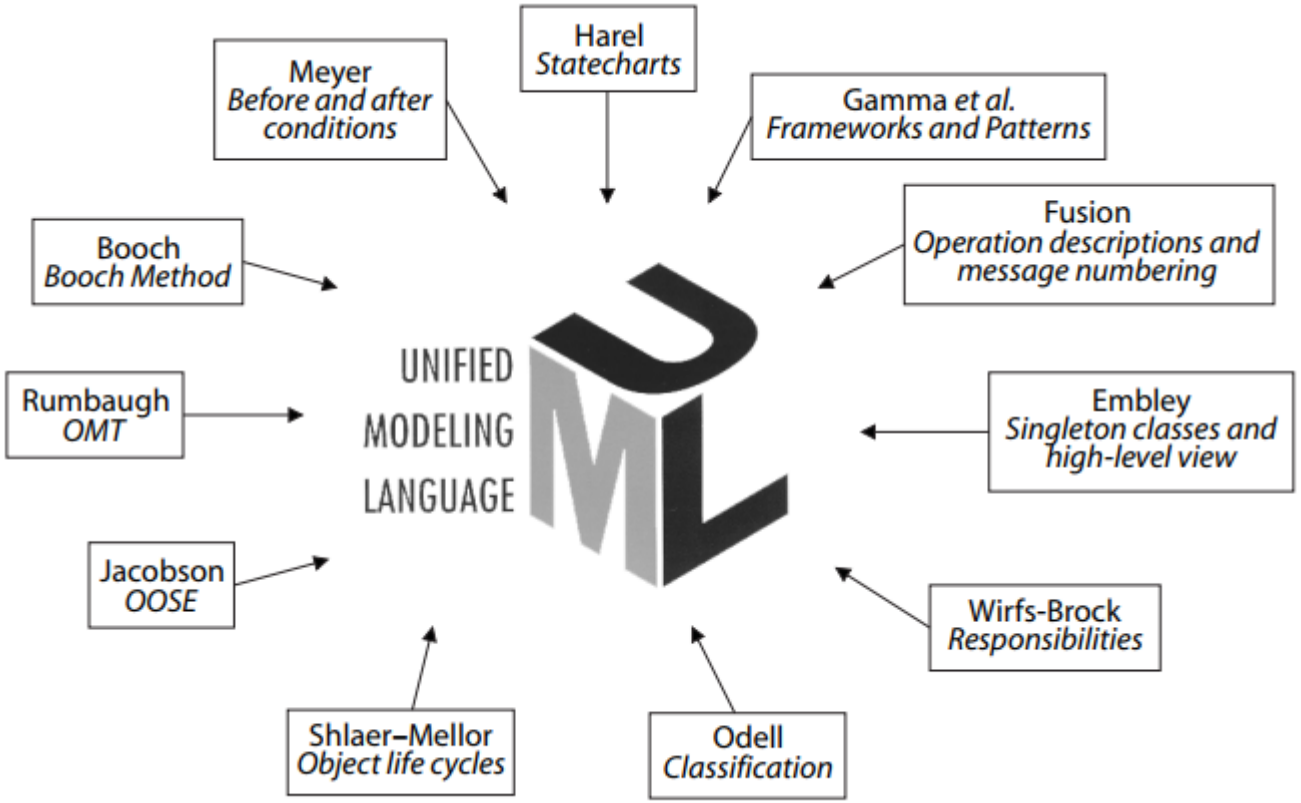
Part - I

Let’s combine UP with UML



Potted history of UML

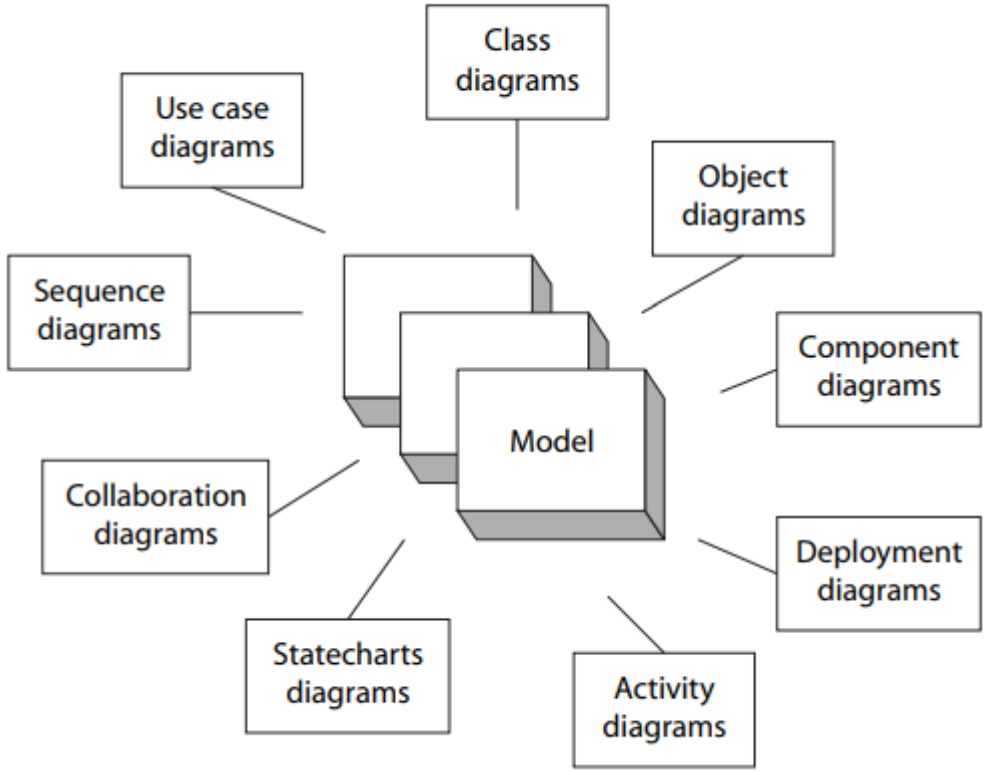
Influences on UML



The primary diagrams that comprise the UML

• Use case diagrams:Essentially, these present the interactions between users (human or otherwise) and the system. They therefore also highlight the primary functionality of the system.
• Class diagrams: These diagrams present the static (class) structure of the system. They are the core of the UML notation and of an object-oriented design.
• Object diagrams: These diagrams use notation which is almost identical to class diagrams, but they present the objects and their relationships at a particular point in time. Object diagrams are not as important as class diagrams, but can be very useful.
• Activity diagrams: These describe the flow of activities or tasks, typically within an operation. They are a bit like a graphical pseudocode.
• Sequence diagrams: These diagrams show the sequence of messages sent between collaborating objects for a particular task. They highlight the flow of control between the objects.
• Collaboration diagrams: These diagrams show the sequence of messages sent between collaborating objects for a particular task. The diagrams highlight the relationships between the collaborating objects. Tools allow you to generate collaboration diagrams from sequence diagrams (and vice versa).
• Statecharts: A statechart, or state diagram, illustrates the states an object can be in and the transitions which move the object between states.
• Component diagrams: These diagrams are used to illustrate the physical structure of the code in terms of the source code. In Java this means the class files and Java Archive Files (JAR),as well as items such as Web Archive Files (WAR) and Enterprise Archive Files (EAR) in the Java 2 Enterprise Edition architecture.
• Deployment diagrams: Deployment diagrams illustrate the physical architecture of the system in terms of processes, networks and the location of components.

Complete System Model



What is UML and What is not

The UML is thus a language for:
- visualizing,
- specifying,
- describing, and
- documenting a software system.

However, the UML is not a design method, it is purely a notation for documenting a design (note that the above all relate to describing the design). A notation on its own is not enough: a method indicating how to apply that design is required. Conceptually the UML can be used with any appropriate object-oriented design method.

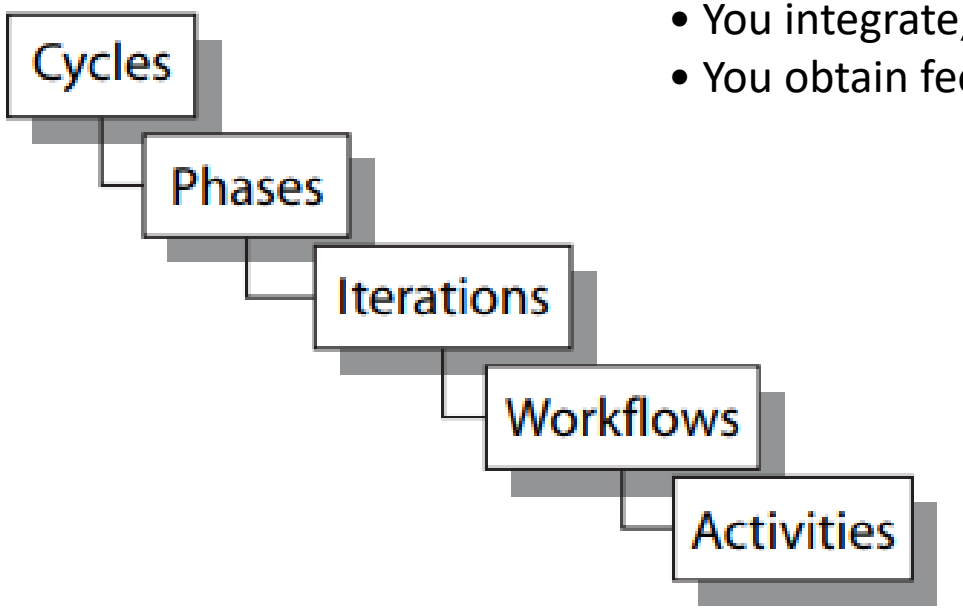
First, it is worth trying to dispel one of the major myths surrounding the UML – it is not a methodology or a process! The UML is a notation that can be used to describe object-oriented systems. This notation tells you nothing about how you should go about carry out a design, producing the elements of a UML diagram or identifying how you should structure your models.

Another myth to deal with is that the UML is complete. It is not!

UML and UP

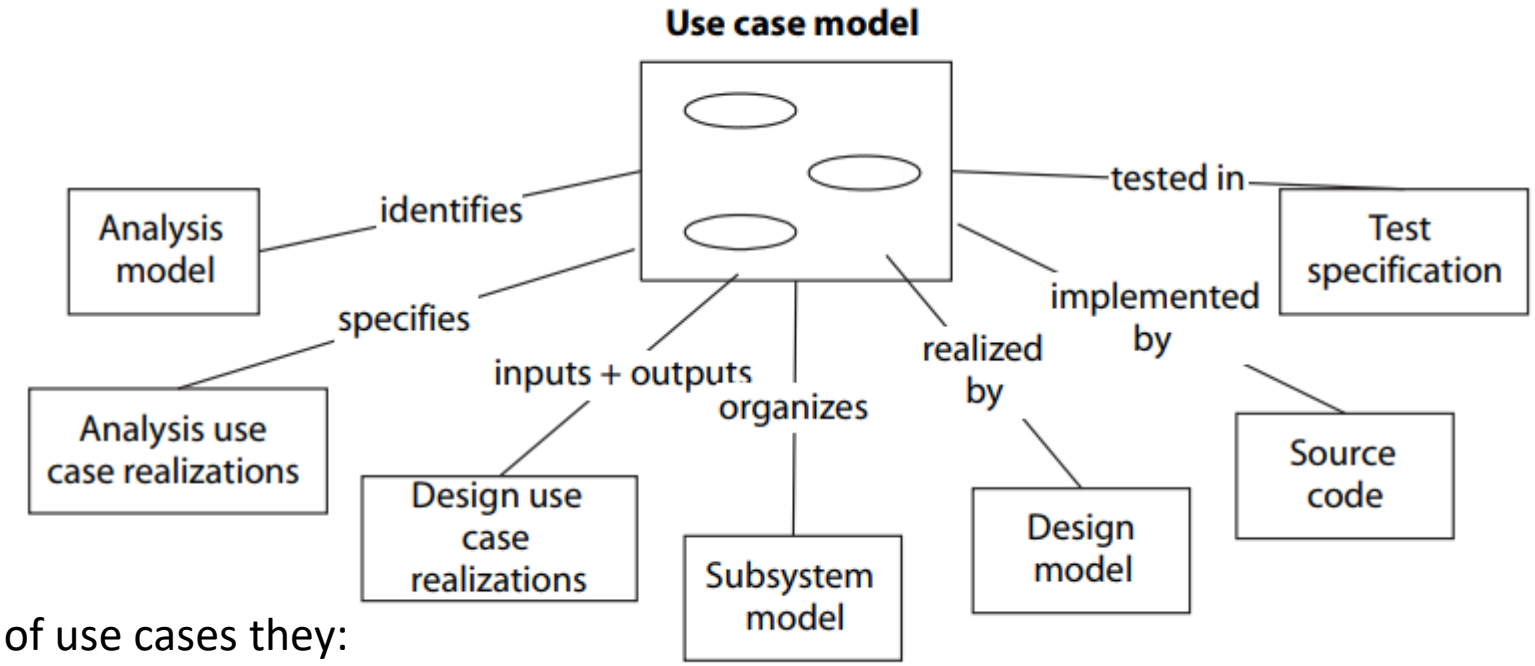
Essentially the following holds with the iterative approach in the Unified Process:

- You plan a little.
- You specify, design and implement a little.
- You integrate, test and run.
- You obtain feedback before next iteration.



Building blocks of Unifed Process

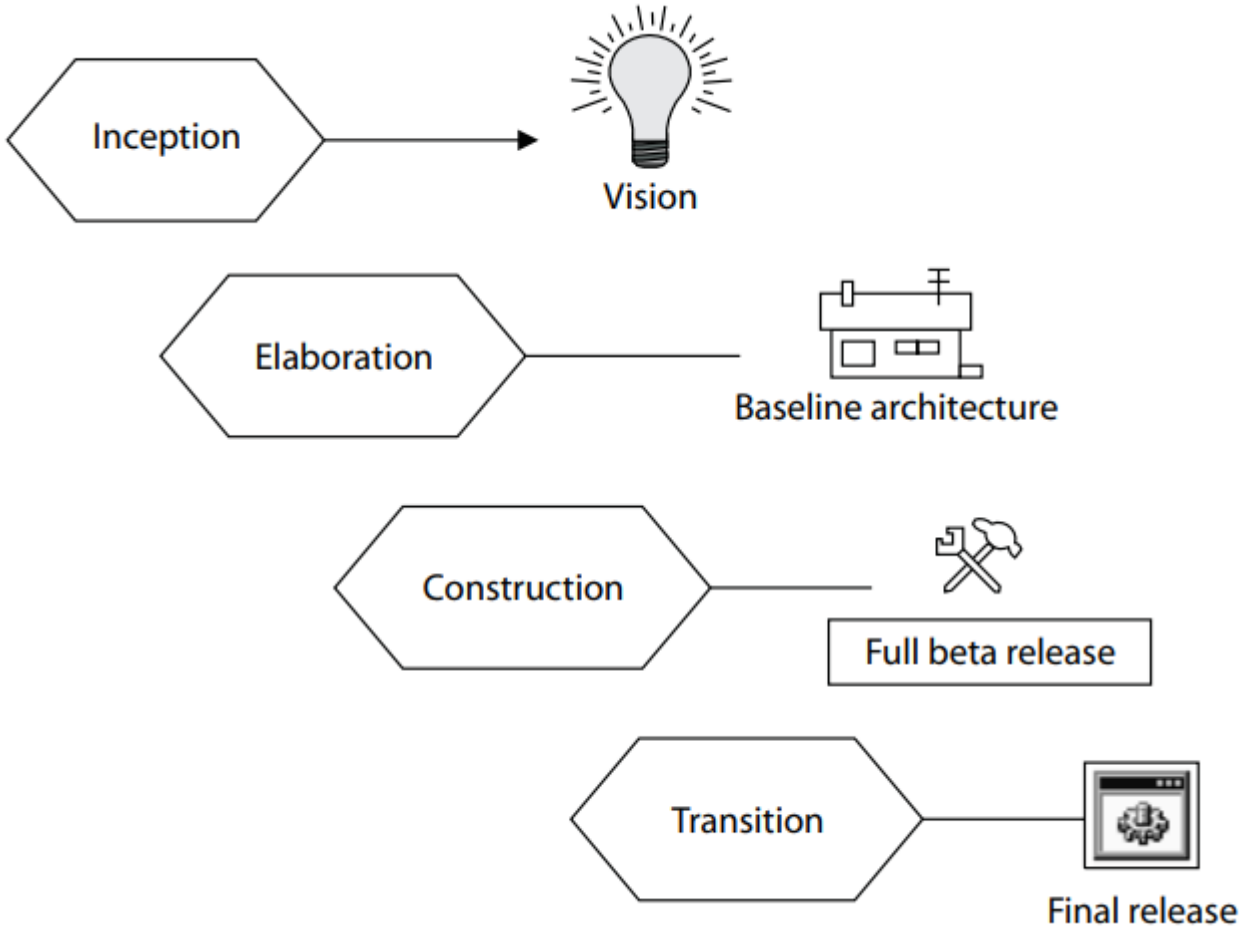
The roles of Use cases



To summarize the role of use cases they:

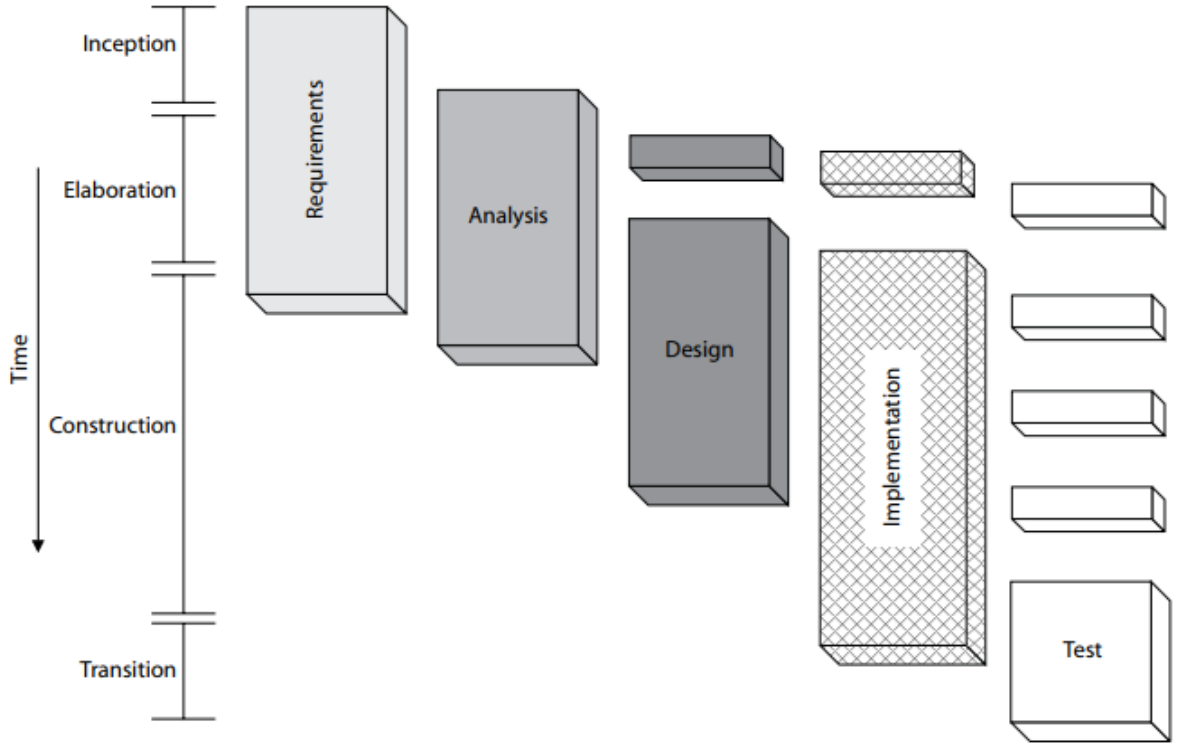
- identify the users of the system and their requirements
- aid in the creation and validation of the system's architecture
- help produce the definition of test cases and procedures
- direct the planning of iterations
- drive the creation of user documentation
- direct the deployment of the system
- synchronize the content of different models
- drive traceability throughout models

Major deliverables of UP's phases



- Inception: The output of this phase is the vision for the system. This includes a very simplified use case model (to identify what the primary functionality of the system is) and a very tentative architecture, and the most important or significant risks are identified and the elaboration phase is planned.
- Elaboration: The primary output of this phase is the architecture, along with a detailed use case model and a set of plans for the construction phase.
- Construction: The end result of this phase is the implemented product which includes the software as well as the design and associated models. The product may not be without defects, as some further work has yet to be completed in the transition phase.
- Transition: he transition phase is the last phase of a cycle. The major milestone met by this phase is the final production-quality release of the system.

Diciplines versus phases



- Requirements:This discipline focuses on the activities which allow the functional and nonfunctional requirements of the system to be identified. The primary product of this discipline is the use case model.

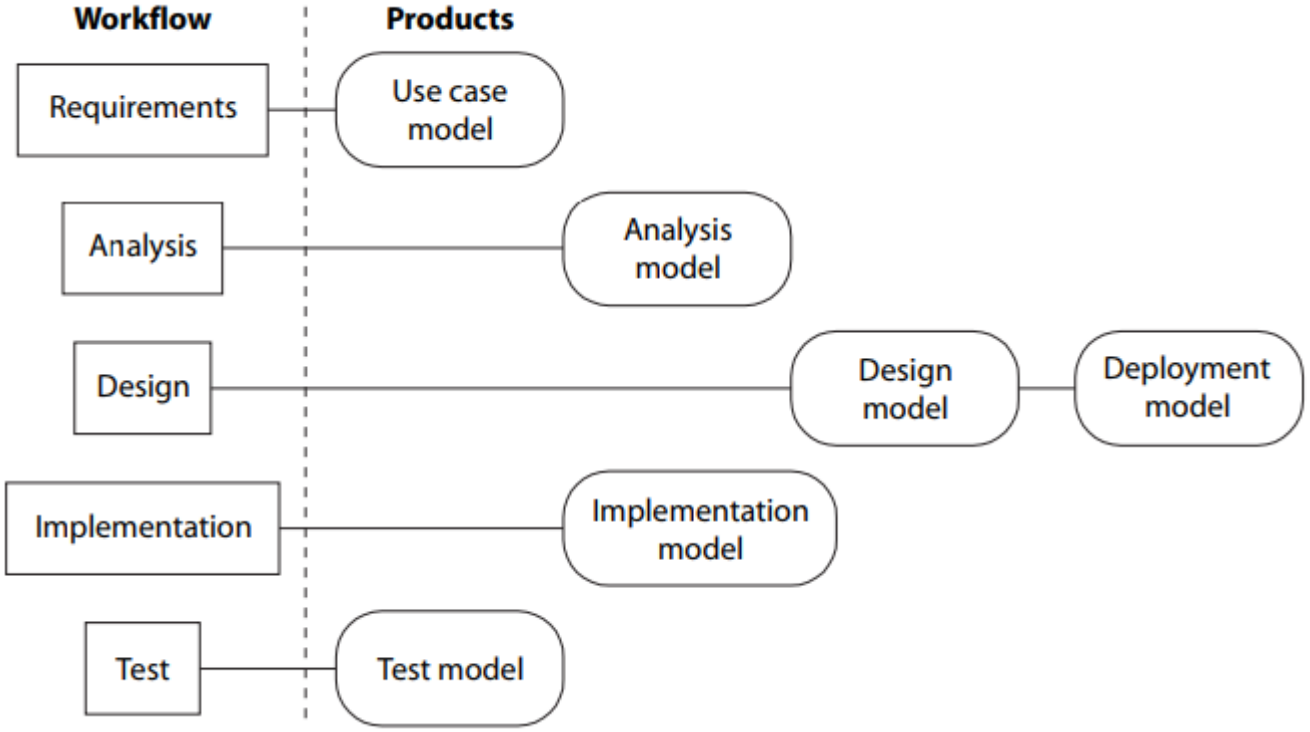
- Analysis: The aim of this discipline is to restructure the requirements identified in the requirements discipline in terms of the software to be built rather than in the user's less precise terms. It can be seen as a first cut at a design; however, that is to miss the point of what this discipline aims to achieve.

- Design: The design discipline produces the detailed design which will be implemented in the next discipline.

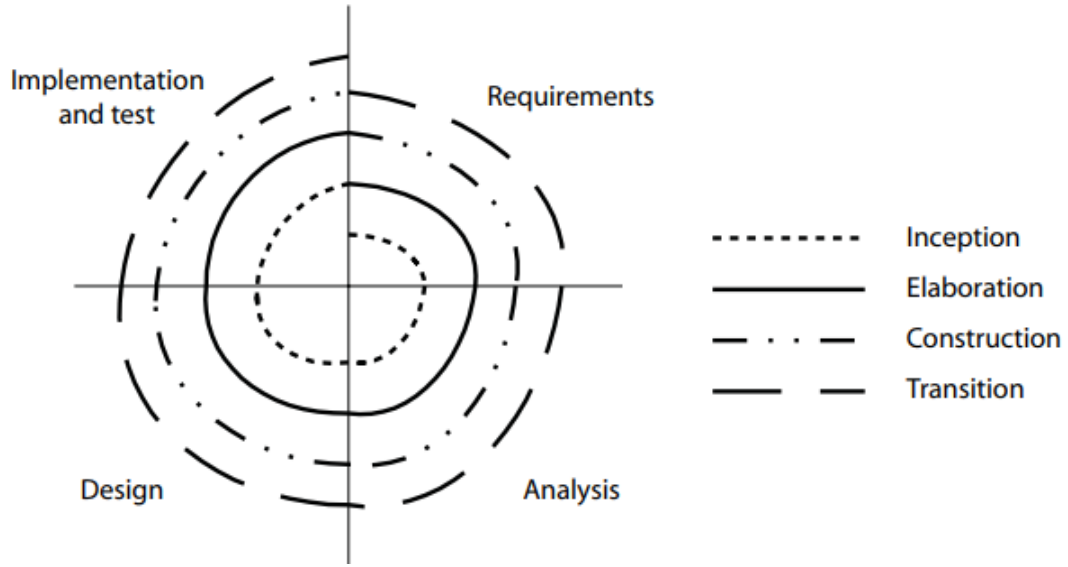
- Implementation: This discipline represents the coding of the design in an appropriate programming language and the compilation, packaging, deployment and documenting of the software.

- Test: The test discipline describes the activities to be carried out to test the software to ensure that it meets the user's requirements, that it is reliable etc.

UP and UML



Product of diciplines



UP is sprial

Diciplines with comprimised activities