UNIFIED PROCESS (UP)

What is the UP?

A software development process describes an approach to building, deploying, and possibly maintaining software. The Unified Process has emerged as a popular iterative software development process for building object-oriented systems. In particular, the Rational Unified Process or RUP a detailed refinement of the Unified Process, has been widely adopted.

The UP combines commonly accepted best practices, such as an **iterative lifecycle and risk-driven development, into a cohesive and well-documented process description.**

UP for three reasons

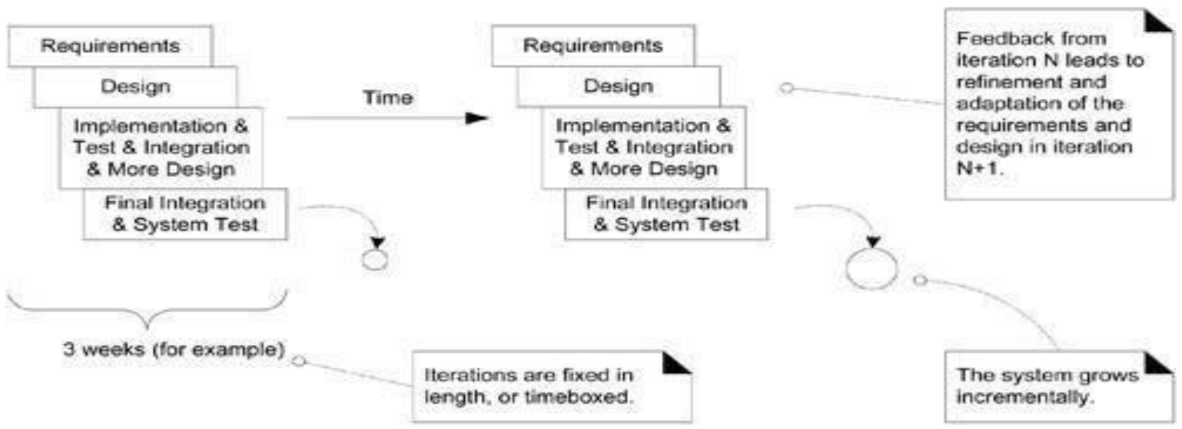
1. The UP is an iterative process.
2. UP practices provide an example structure for how to do and thus how to explain OOA/D.
3. The UP is flexible, and can be applied in a lightweight and agile approach that includes practices from other agile methods

Iterative and Evolutionary Development

A key practice in both the UP and most other modern methods is iterative development.

- In this lifecycle approach, development is organized into a series of short, fixed-length (for example, three-week) mini-projects called iterations; the outcome of each is a tested, integrated, and executable partial system. Each iteration includes its own requirements analysis, design, implementation, and testing activities.
- The system grows incrementally over time, iteration by iteration, and thus this approach is also known as iterative and incremental development . Because feedback and adaptation evolve the specifications and design, it is also known as iterative and evolutionary development.

Iterative and evolutionary development.

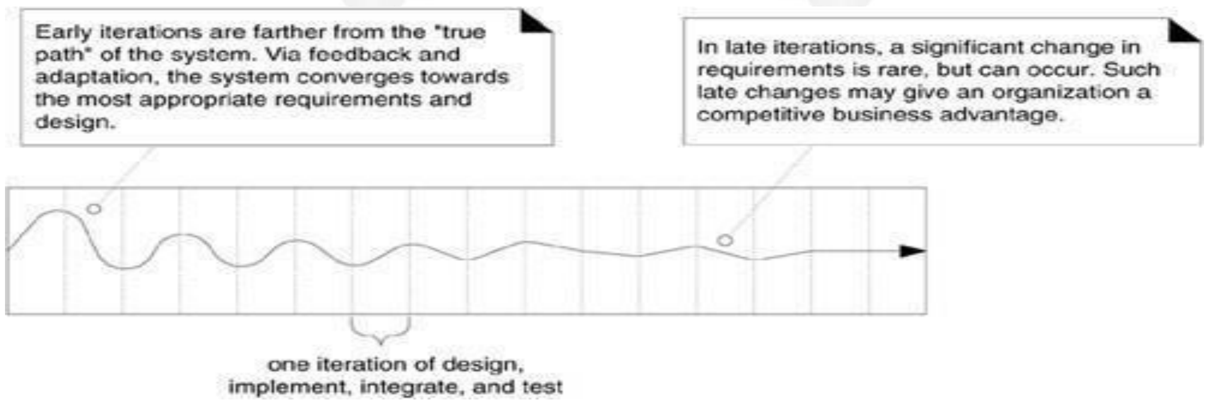

Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
Time
Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
Feedback from iteration N leads to refinement and adaptation of the requirements and design in iteration N+1.
3 weeks (for example)
Iterations are fixed in length, or timeboxed.
The system grows incrementally.

Notice in this example that there is neither a rush to code, nor a long drawn-out design step that attempts to perfect all details of the design before programming. The system may not be eligible for production deployment until after many iterations; for example, 10 or 15 iterations.

To Handle Change on an Iterative Project

- Each iteration involves choosing a small subset of the requirements, and quickly designing, implementing, and testing
- In addition to requirements clarification, activities such as load testing will prove if the partial design and implementation are on the right path, or if in the next iteration, a change in the core architecture is required.

Iterative feedback and evolution leads towards the desired system. The requirements and design instability lowers over time.


Early iterations are farther from the "true path" of the system. Via feedback and adaptation, the system converges towards the most appropriate requirements and design.
In late iterations, a significant change in requirements is rare, but can occur. Such late changes may give an organization a competitive business advantage.
one iteration of design, implement, integrate, and test

- Work proceeds through a series of structured build-feedback-adapt cycles. in early iterations the deviation from the "true path" of the system (in terms of its final requirements and design) will be larger than in later iterations. Over time, the system converges towards this path, as illustrated in Figure

Benefits of Iterative Development

- less project failure, better productivity, and lower defect rates; shown by research into iterative and evolutionary methods
- early rather than late mitigation of high risks (technical, requirements, objectives, usability, and so forth)
- early visible progress
- early feedback, user engagement, and adaptation, leading to a refined system that more closely meets the real needs of the stakeholders
- managed complexity; the team is not overwhelmed by "analysis paralysis" or very long and complex steps
- the learning within an iteration can be methodically used to improve the development process itself, iteration by iteration

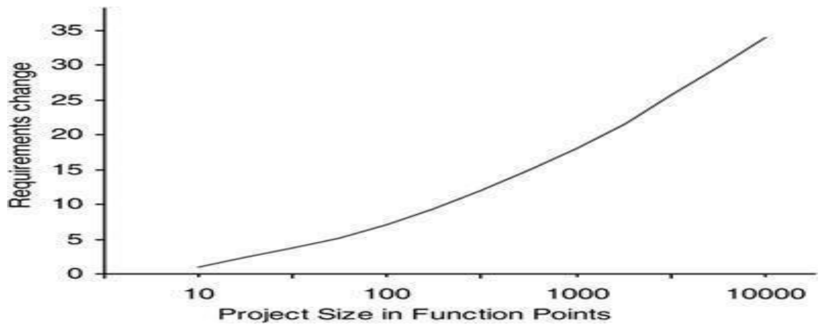
What is Iteration Time boxing?

- Most iterative methods recommend an iteration length between two and six weeks.
- Small steps, rapid feedback, and adaptation are central ideas in iterative development; long iterations subvert the core motivation for iterative development and increase project risk.
- A very long time-boxed iteration misses the point of iterative development. Short is good.

Iterations are time-boxed, or fixed in length. For example, if the next iteration is chosen to be three weeks long, then the partial system must be integrated, tested, and stabilized by the scheduled date-date slippage is illegal. If it seems that it will be difficult to meet the deadline, the recommended response is to de-scope-remove tasks or requirements from the iteration, and include them in a future iteration, rather than slip the completion date.

In a waterfall lifecycle process there is an attempt to define all or most of the requirements before programming. It is strongly associated with

- high rates of failure
- lower productivity
- higher defect rates



Percentage of change on software projects of varying sizes.

The Need for Feedback and Adaptation

In complex, changing systems feedback and adaptation are key ingredients for success.

- Feedback from early development, programmers trying to read specifications, and client demos to refine the requirements.
- Feedback from tests and developers to refine the design or models.
- Feedback from the progress of the team tackling early features to refine the schedule and estimates.
- Feedback from the client and marketplace to re-prioritize the features to tackle in the next iteration.

What is Risk-Driven and Client-Driven Iterative Planning?

The UP encourages a combination of risk-driven and client-driven iterative planning. This means that the goals of the early iterations are chosen to 1) identify and drive down the highest risks, and 2) build visible features that the client cares most about.Risk-driven iterative development includes more specifically the practice of architecture-centric iterative development, i.e early iterations focus on building, testing, and stabilizing the core architecture

What are Agile Methods and Attitudes?

Agile development methods usually apply time boxed iterative and evolutionary development, employ adaptive planning, promote incremental delivery, and include other values and practices that encourage agility rapid and flexible response to change.

The Agile Manifesto and Principles

The Agile Manifesto

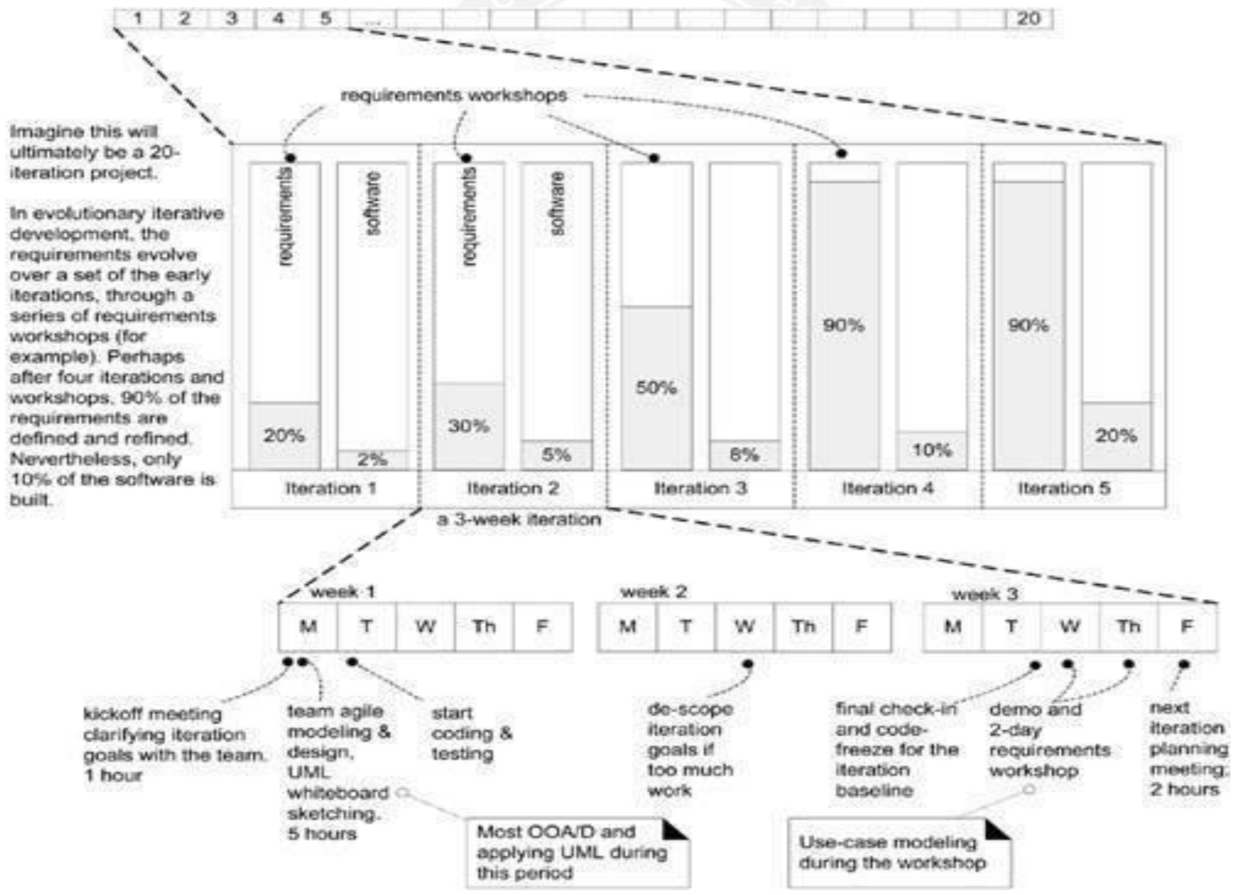
Individuals and interactions	over processes and tools
Working software	over comprehensive documentation
Customer collaboration	over contract negotiation
Responding to change	over following a plan

The Agile 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 time scale.

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.
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.
9. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
10. Continuous attention to technical excellence and good design enhances agility
11. Simplicity the art of maximizing the amount of work not done is essential
12. The best architectures, requirements, and designs emerge from self-organizing teams.13. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

This example assumes there will ultimately be 20 iterations on the project before delivery:


1 2 3 4 5 ... 20
requirements workshops
Imagine this will ultimately be a 20-iteration project.
In evolutionary iterative development, the requirements evolve over a set of the early iterations, through a series of requirements workshops (for example). Perhaps after four iterations and workshops, 90% of the requirements are defined and refined. Nevertheless, only 10% of the software is built.
requirements
software
20%
2%
30%
5%
50%
8%
90%
10%
90%
20%
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
a 3-week iteration
week 1
week 2
week 3
M T W Th F
kickoff meeting clarifying iteration goals with the team. 1 hour
team agile modeling & design, UML whiteboard sketching. 5 hours
start coding & testing
de-scope iteration goals if too much work
final check-in and code-freeze for the iteration baseline
demo and 2-day requirements workshop
next iteration planning meeting. 2 hours
Most OOA/D and applying UML during this period
Use-case modeling during the workshop

Evolutionary analysis and design the majority in early iterations.

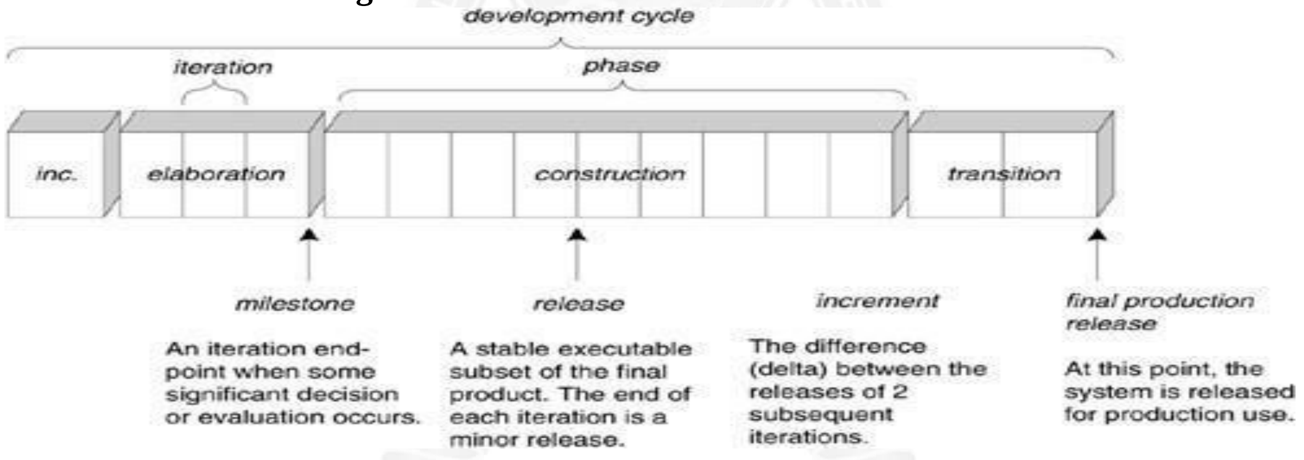
UP Phases

A UP project organizes the work and iterations across four major phases:

1. **Inception -** approximate vision, business case, scope, vague estimates.
2. **Elaboration -** refined vision, iterative implementation of the core architecture, resolution of high risks, identification of most requirements and scope, more realistic estimates.
3. **Construction -** iterative implementation of the remaining lower risk and easier elements, and preparation for deployment.
4. **Transition -** beta tests, deployment.

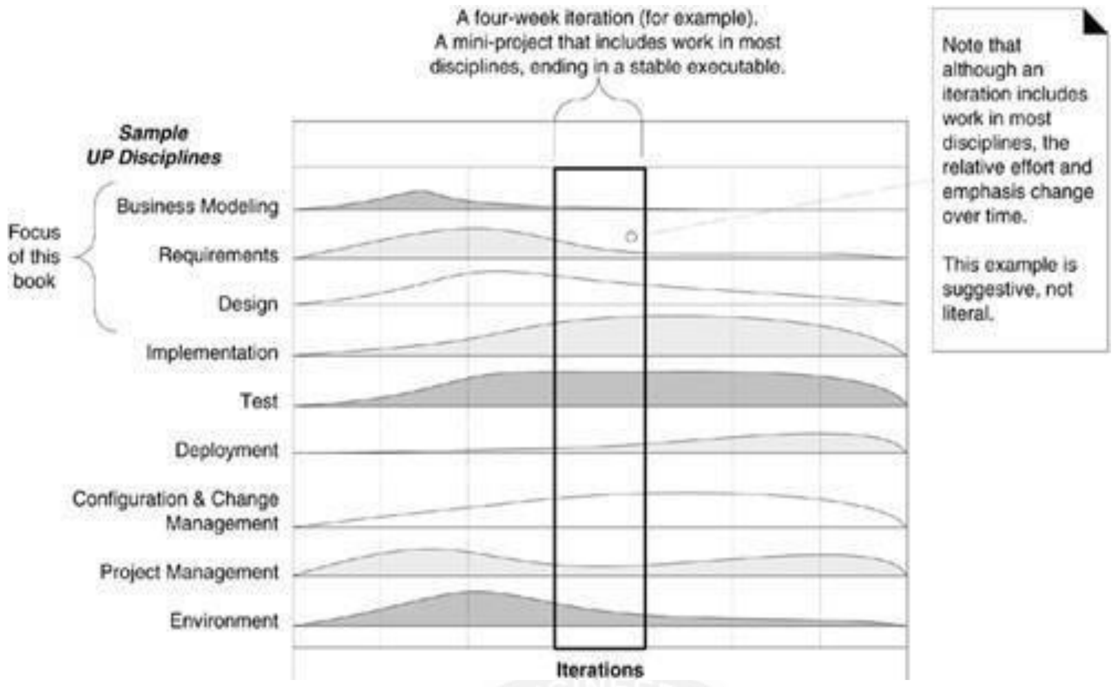
This is not the old "waterfall" or sequential lifecycle of first defining all the requirements, and then doing all or most of the design. Inception is not a requirements phase; rather, it is a feasibility phase, where investigation is done to support a decision to continue or stop.Similarly, elaboration is a phase where the core architecture is iteratively implemented, and high-risk issues are mitigated.

Figure - Schedule-oriented terms in the UP.


development cycle
iteration
phase
inc.
elaboration
construction
transition
milestone
release
increment
final production release
An iteration end-point when some significant decision or evaluation occurs.
A stable executable subset of the final product. The end of each iteration is a minor release.
The difference (delta) between the releases of 2 subsequent iterations.
At this point, the system is released for production use.

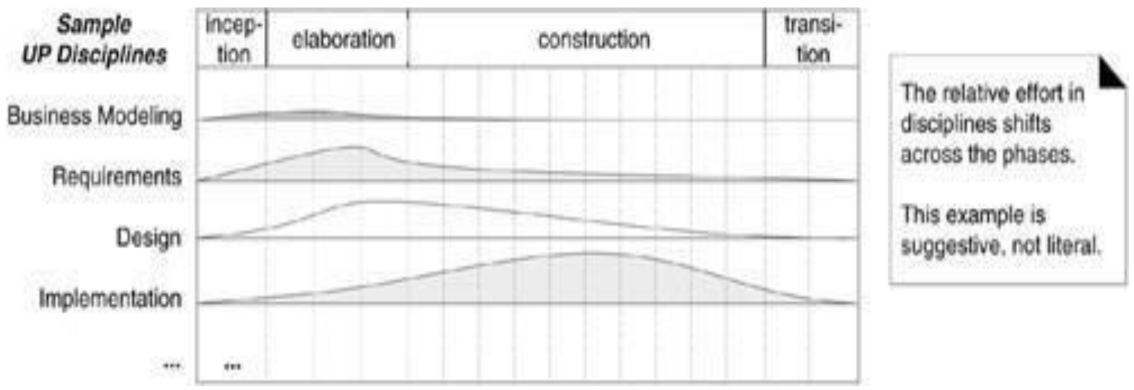
The UP Disciplines

Disciplines a set of activities (and related artifacts) in one subject area, such as the activities within requirements analysis. In the UP, an artifact is the general term for any work product: code, Web graphics, database schema, text documents, diagrams, models, and so on. There are several disciplines in the UP:



- **Business Modeling -** The Domain Model artifact, to visualize noteworthy concepts in the application domain.
- **Requirements -** The Use-Case Model and Supplementary Specification artifacts to capture functional and non-functional requirements.
- **Design -** The Design Model artifact, to design the software objects

What is the Relationship Between the Disciplines and Phases?



Definition: the Development Case: The choice of practices and UP artifacts for a project may be written up in a short document called the Development Case (an artifact in the
Environment discipline).

Discipline	Practice	Artifact	Incep.	Elab.	Const.	Trans.
		Iteration	I1	E1..En	C1..Cn	T1..T2
Business Modeling	agile modeling req. workshop	Domain Model		s		
Requirements	req. workshop vision box exercise dot voting	Use-Case Model	s	r		
		Vision	s	r		
		Supplementary Specification	s	r		
		Glossary	s	r		
Design	agile modeling test-driven dev.	Design Model		s	r	
		SW Architecture Document		s		
		Data Model		s	r	

Implementation	test-driven dev. pair programming continuous integration coding standards	…				
Project Management	agile PM daily Scrum meeting	…				
…						

Table- Sample Development Case. s - start; r - refine