Lecture 6

Unified Process

Agenda

- Define an iterative and adaptive process.
- Define fundamental concepts in the Unified Process.

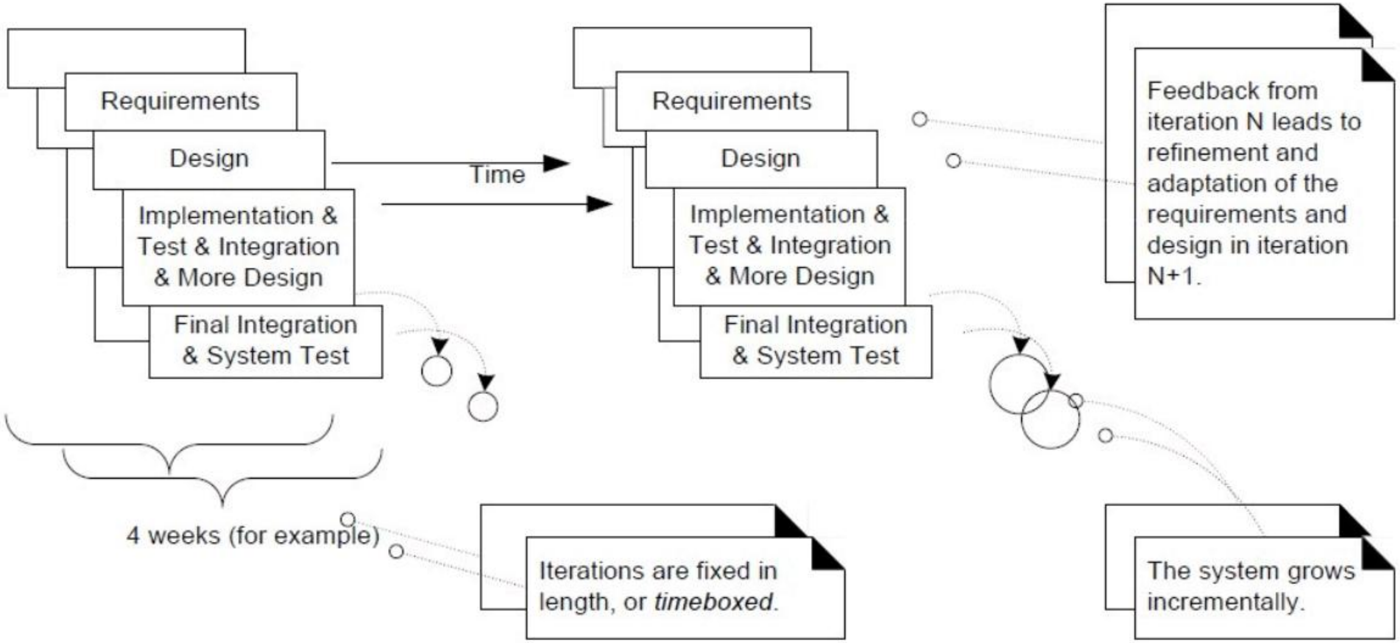
UP

- The **Unified Process** has emerged as a popular software development process for building object-oriented systems.
- The Unified Process (UP) combines commonly accepted best practices, such as an iterative lifecycle and risk-driven development, into a cohesive and well-documented description.

Important Idea

- The iterative lifecycle is based on the successive enlargement and refinement of a system through multiple iterations, with cyclic feedback and adaptation as core drivers to converge upon a suitable system.
- The system grows incrementally over time, iteration by iteration, and thus this approach is also known as **iterative and incremental development**

Iterative and incremental development.



Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
Time
Requirements
Design
Implementation & Test & Integration & More Design
Final Integration & System Test
4 weeks (for example)

Feedback from iteration N leads to refinement and adaptation of the requirements and design in iteration N+1.

Iterations are fixed in length, or *timeboxed*.

The system grows incrementally.

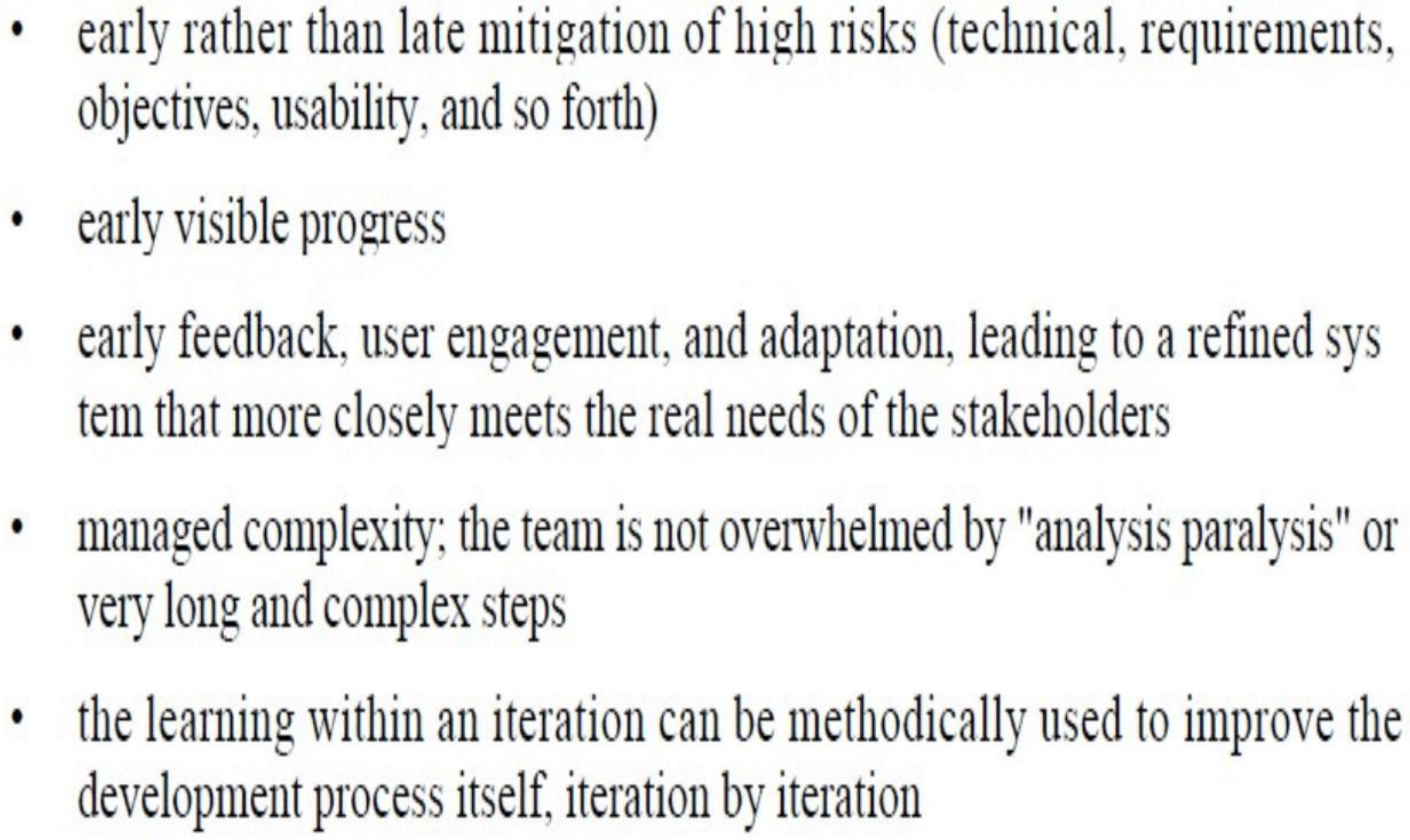
Highlights

- 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.
- A "little" forethought regarding the design with visual modeling using rough and fast UML drawings is done; perhaps a half or full day by developers doing design work in pairs.
- The result of each iteration is an executable but incomplete system; it is not ready to deliver into production. The system may not be eligible for production deployment until after many iterations; for example, 10 or 15 iterations.

Contd..

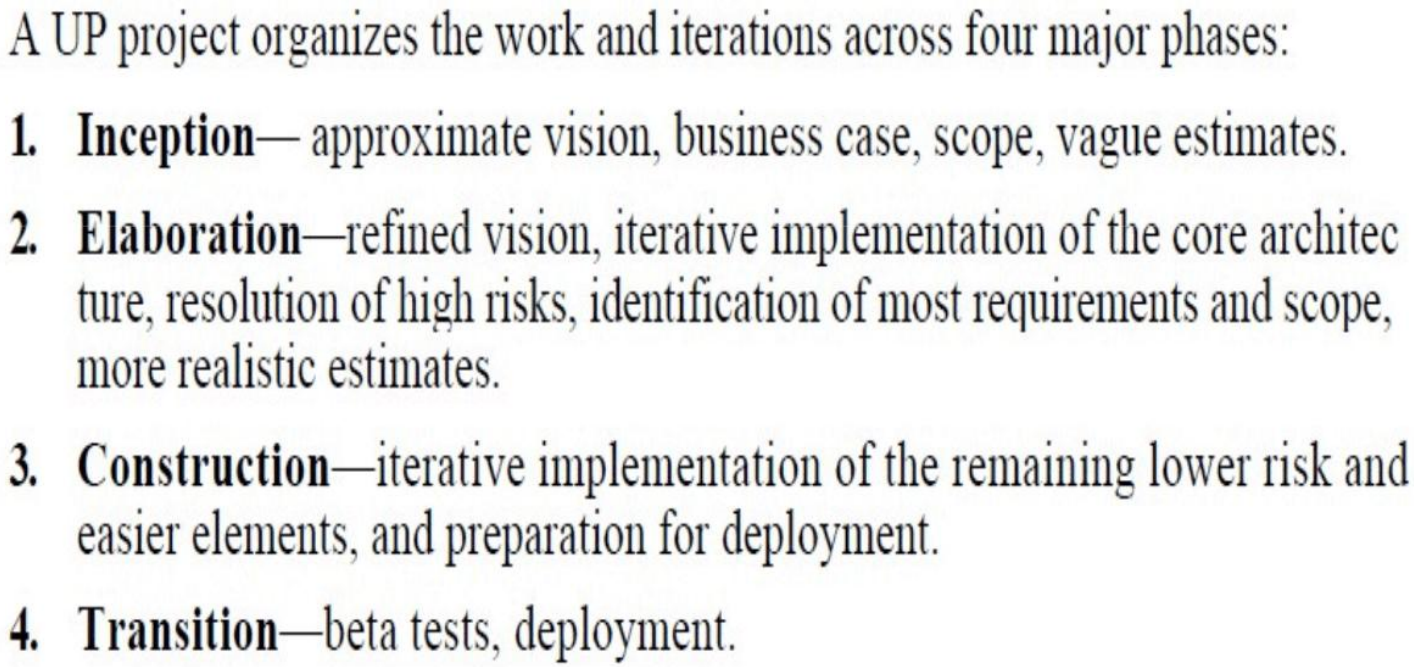
- The output of an iteration is *not an experimental or throw-away prototype, and* iterative development is not prototyping.
- Rather, the output is a production-grade subset of the final system.
- each iteration tackles new requirements and incrementally extends the system, an iteration may occasionally revisit existing software and improve it;

Benefits of Iterative Development



- 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 sys tem 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

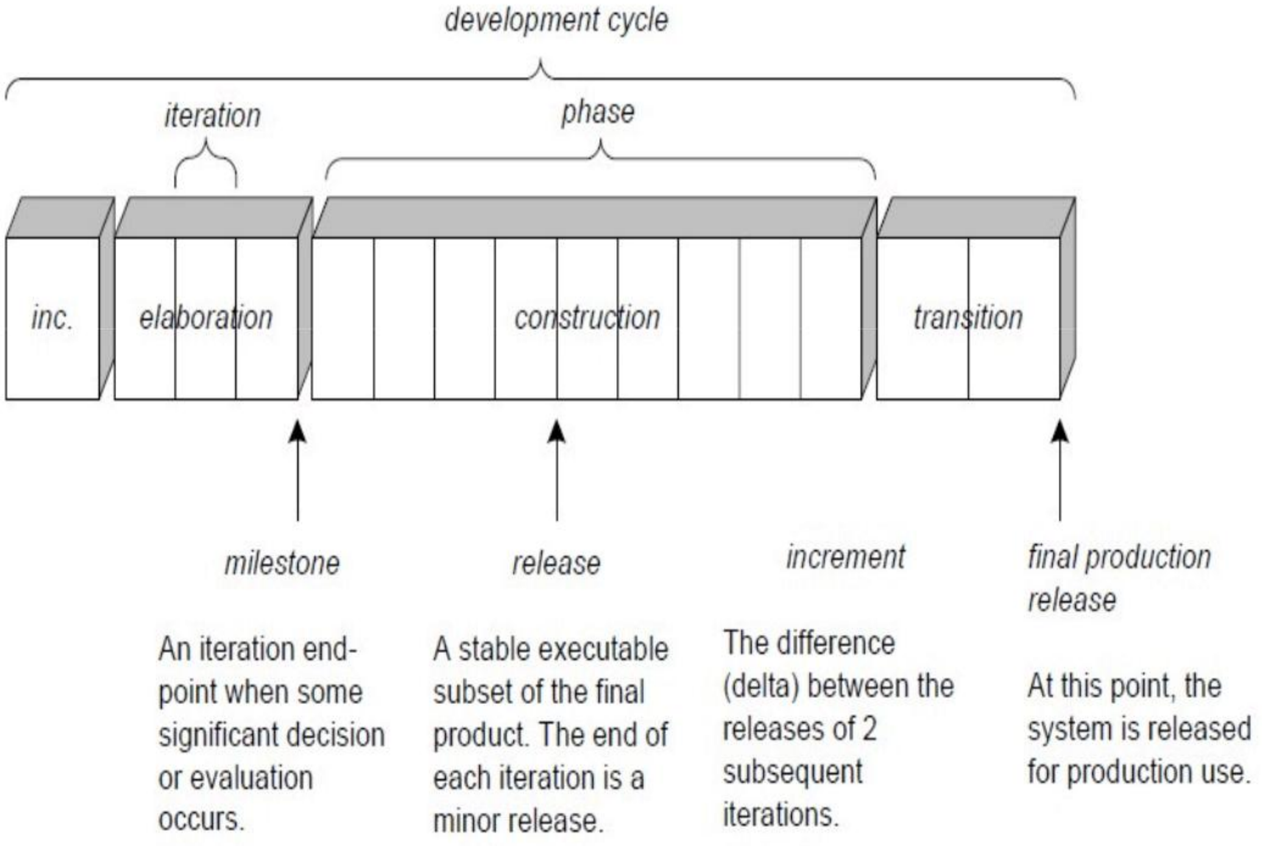
Phases



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.

Schedule-oriented terms in the UP.



development cycle

iteration

phase

inc.

elaboration

construction

transition

milestone

An iteration end-point when some significant decision or evaluation occurs.

release

A stable executable subset of the final product. The end of each iteration is a minor release.

increment

The difference (delta) between the releases of 2 subsequent iterations.

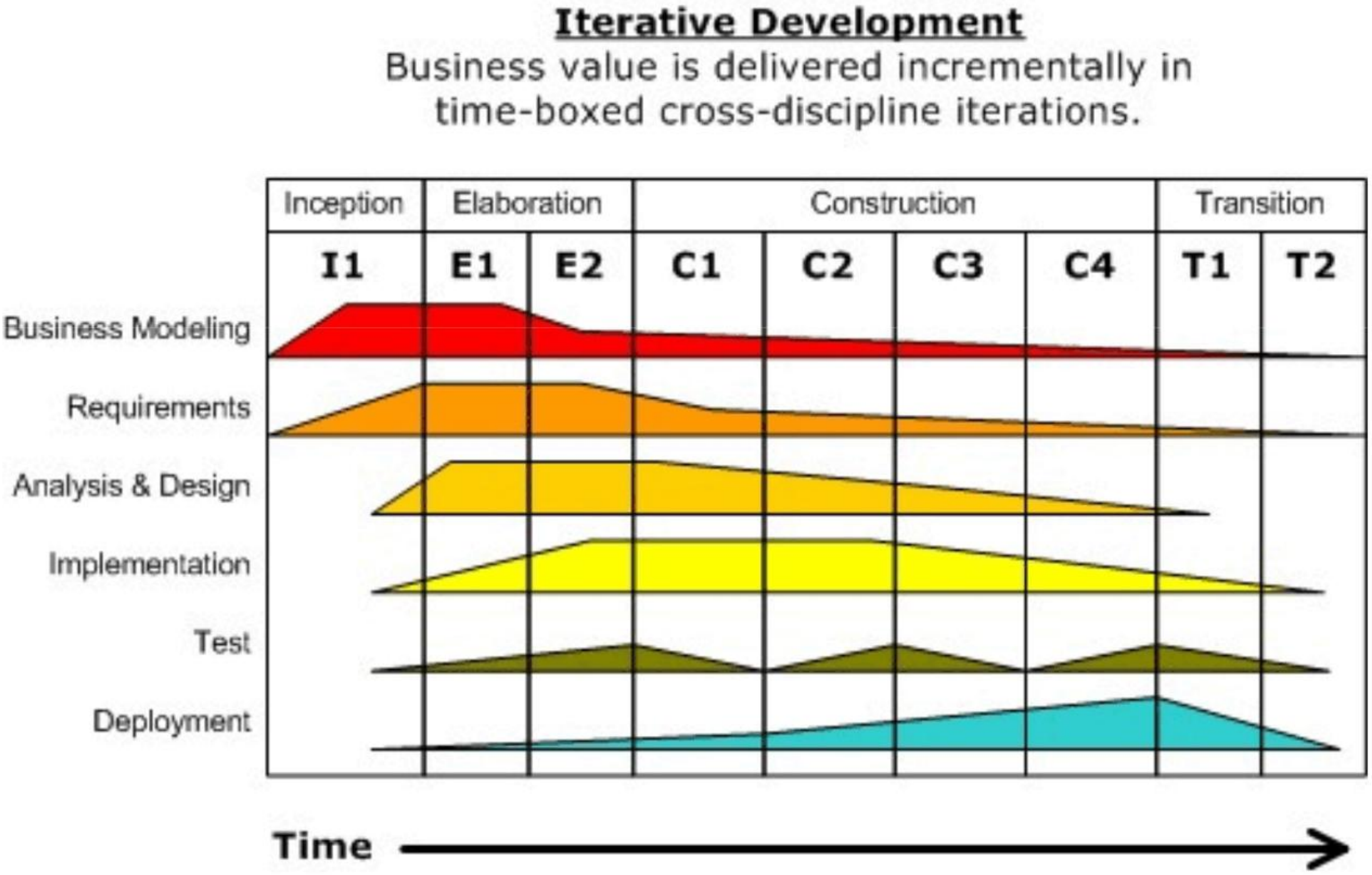
final production release

At this point, the system is released for production use.

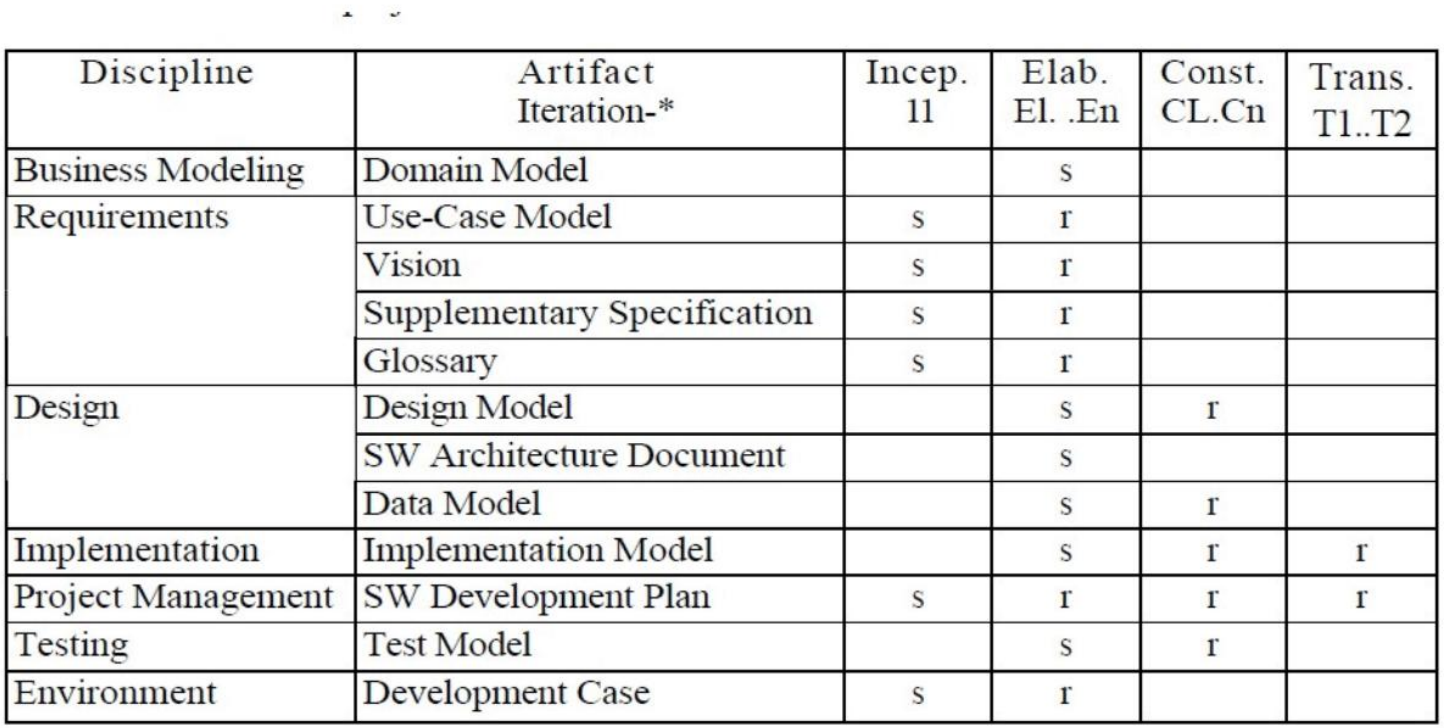
The UP Disciplines (was Workflows)

- The UP describes work activities, such as writing a use case, within **disciplines** (originally called **workflows).**
- 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.

Disciplines and phases



Sample Development Case of UP artifacts, s - start; r - refine



Discipline	Artifact Iteration-*	Incep. 11	Elab. El. .En	Const. CL.Cn	Trans. T1..T2
Business Modeling	Domain Model		s		
Requirements	Use-Case Model	s	r		
	Vision	s	r		
	Supplementary Specification	s	r		
	Glossary	s	r		
Design	Design Model		s	r	
	SW Architecture Document		s		
	Data Model		s	r	
Implementation	Implementation Model		s	r	r
Project Management	SW Development Plan	s	r	r	r
Testing	Test Model		s	r	
Environment	Development Case	s	r		

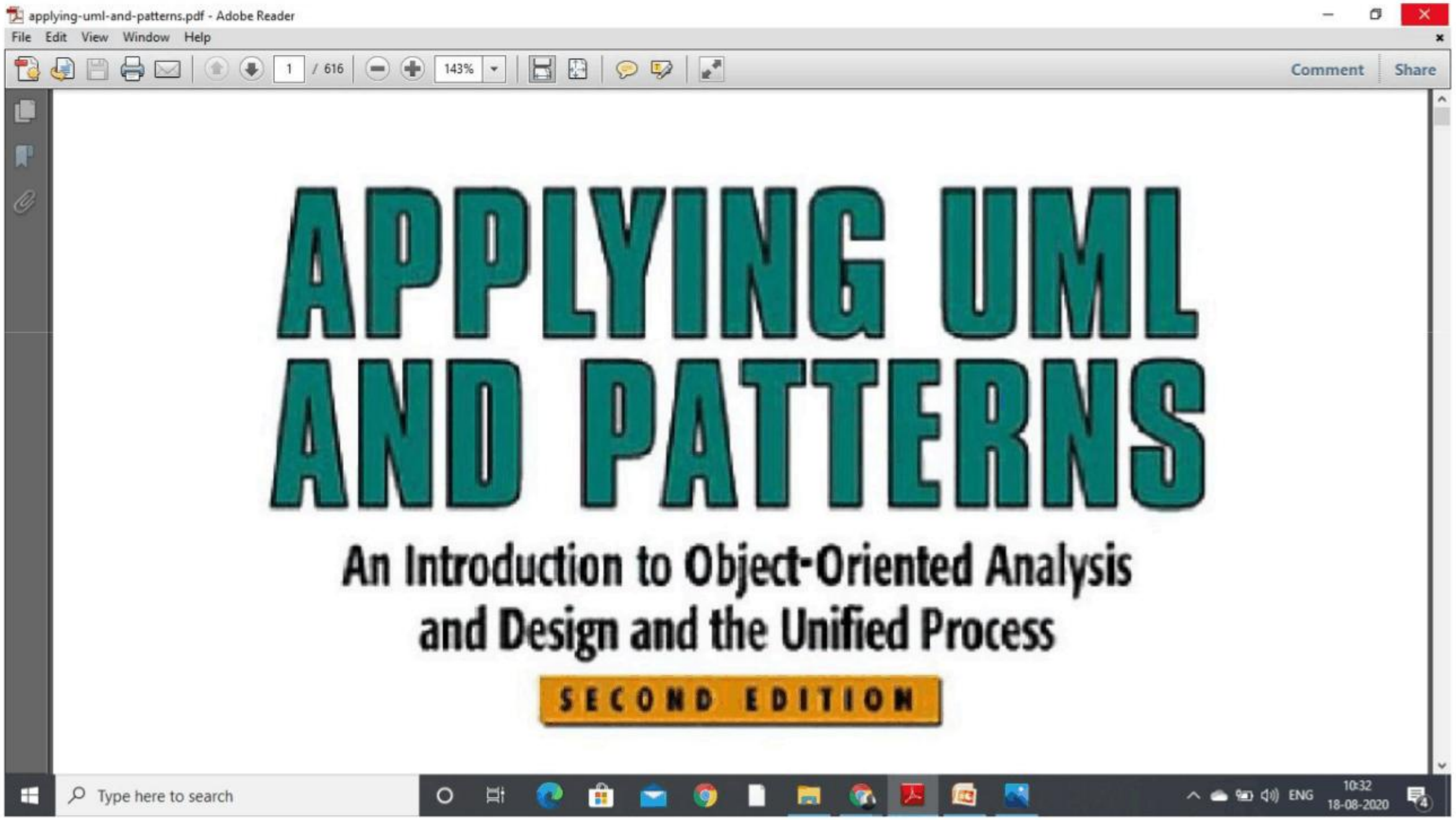
You Know You Didn't Understand the UP When...

- You think that inception = requirements, elaboration = design, and construction = implementation (that is, superimposing a waterfall lifecycle on to the UP).
- You think that the purpose of elaboration is to fully and carefully define models, which are translated into code during construction.
- You try to define most of the requirements before starting design or implementation.

Contd..

- You try to define most of the design before starting implementation; you try to fully define and commit to an architecture before iterative programming and testing.
- A "long time" is spent doing requirements or design work before programming starts.
- You think UML diagramming and design activities are a time to fully and accurately define designs and models in great detail, and of programming as a simple mechanical translation of these into code.

References



APPLYING UML AND PATTERNS

An Introduction to Object-Oriented Analysis and Design and the Unified Process

SECOND EDITION