UNIFIED PROCESS

WHAT IS THE UNIFIED PROCESS

A popular iterative modern process model (framework) derived from the work on the UML and associated process.

The leading object-oriented methodology for the development of large-scale software

Maps out when and how to use the various UML techniques

THE MOST IMPORTANT UP IDEA: ITERATIVE DEVELOPMENT

- The UP(unified process) promotes several best practices, but one stands above the others: **iterative development.**

- In this approach, development is organized into a series of short, fixed-length (for example, four week) mini-projects called **iterations.**

- The outcome of each is a tested, integrated, into executable system.

- Each iteration includes its own requirements analysis, design, implementation, and testing activities.

ITERATIVE AND INCREMENTAL DEVELOPMENT

- The system grows incrementally over time, iteration by iteration, and thus this approach is also known as **iterative and incremental development.**
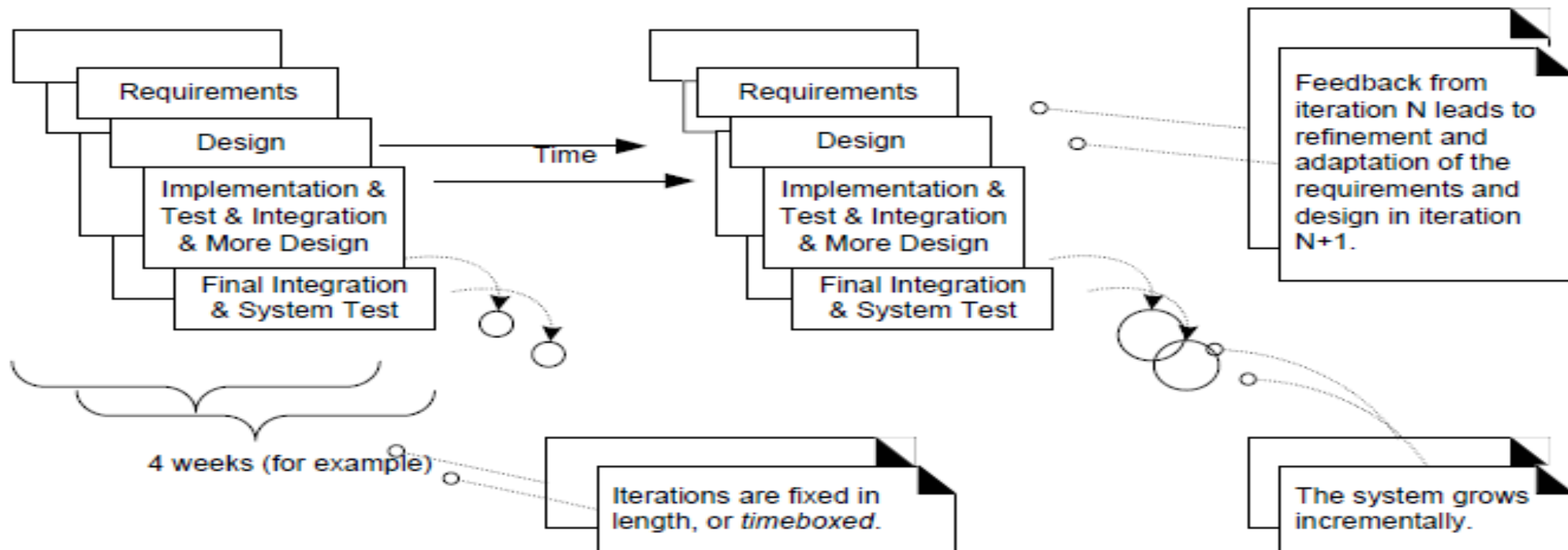
ITERATION RESULT..

- The result of each iteration is an executable but incomplete system.

- 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.

ITERATIVE FEEDBACK AND ADAPTATION LEADS TOWARDS THE DESIRED SYSTEM.

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.
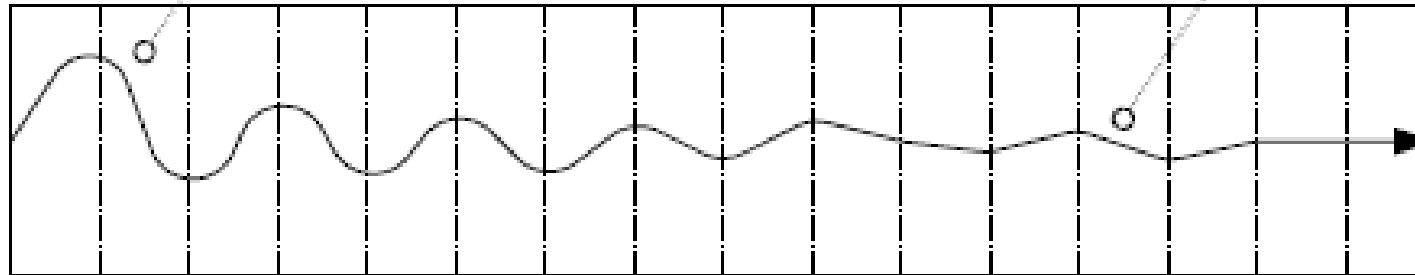
one iteration of design, implement, integrate, and test

BENEFITS OF ITERATIVE DEVELOPMENT

It includes:

- 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

ITERATION LENGTH

- The UP (and experienced iterative developers) recommends an iteration length between two and six weeks.

- Much less than two weeks, and it is difficult to complete sufficient work to get meaningful throughput and feedback;

- Much more than six or eight weeks, and the complexity becomes rather overwhelming, and feedback is delayed.

- A very long iteration misses the point of iterative development.

- Short is good.

TIME-BOXED

- A key idea is that iterations are **time boxed,** or fixed in length.

- For example, if the next iteration is chosen to be four weeks long, then the partial system should be integrated, tested, and stabilized by the scheduled **date—date slippage is discouraged.**

- If it seems that it will be difficult to meet the deadline, the recommended response is to remove tasks or requirements from the iteration, and include them in a future iteration, rather than slip the completion date.

THE 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.

IMP POINT ABOUT PHASES.

- 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 kind of feasibility phase, where just enough investigation is done to support a decision to continue or stop.

- Similarly, elaboration is not the requirements or design phase; rather, it is a phase where the core architecture is iteratively implemented, and high risk issues are mitigated.
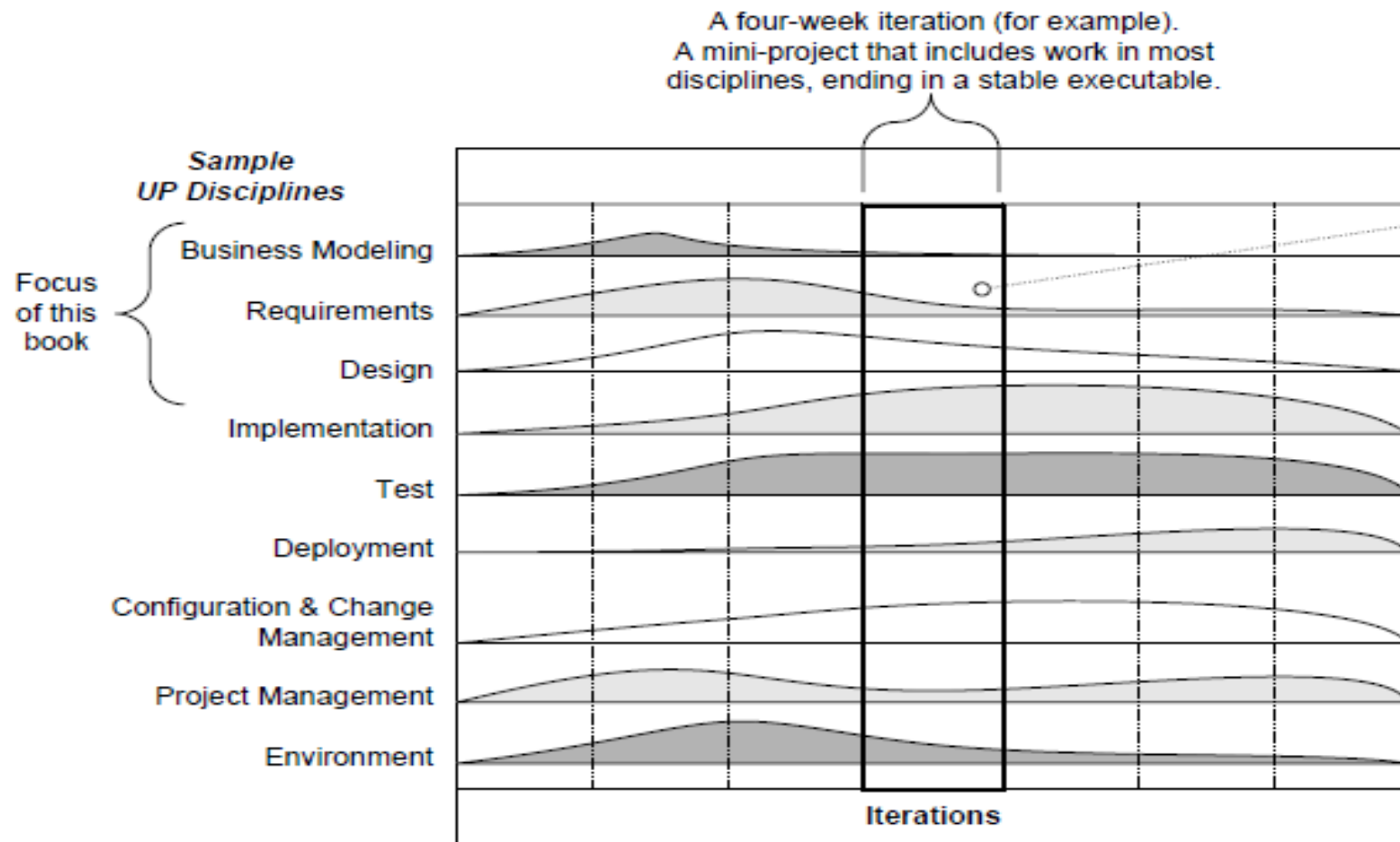
SCHEDULE-ORIENTED TERMS IN THE UP.
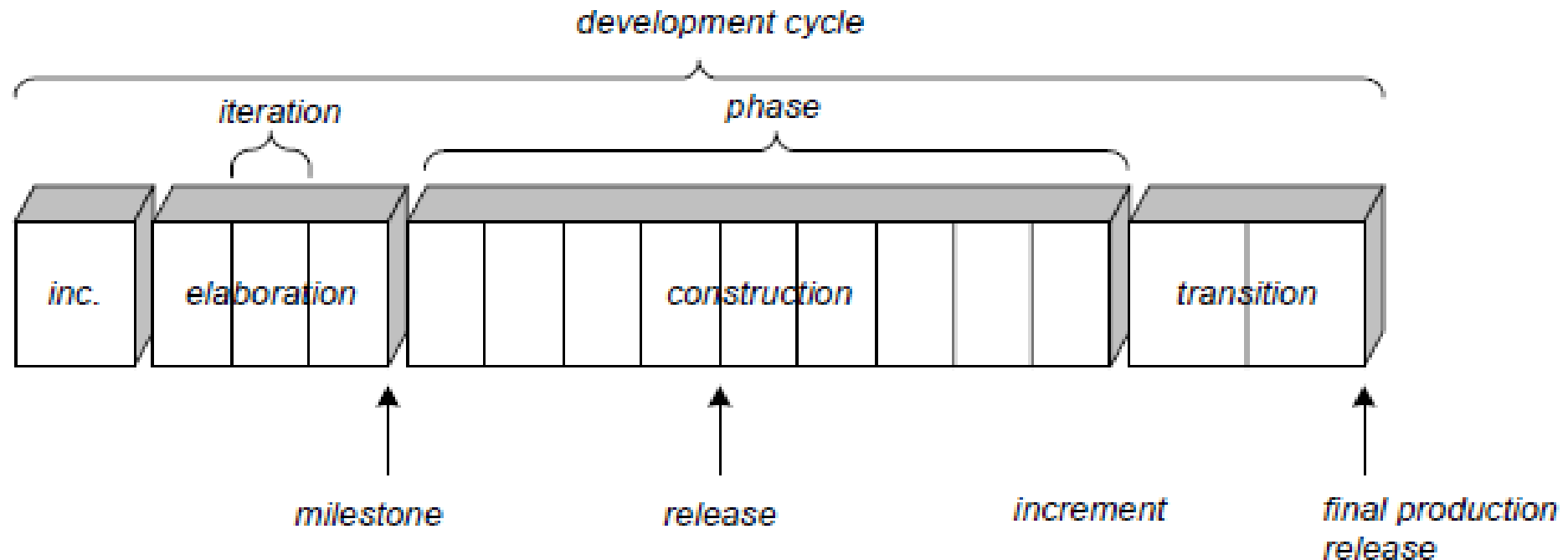
UP DISCIPLINES AS RESOURCE HISTOGRAM