

Introduction to the Unified Process



Iterative, incremental development and your project

You now have some understanding of the advantages of the iterative and incremental approach to development.

But you need to know more about your project's process framework to understand how you will proceed, what you will produce at each stage and **why**.

You will use the **Unified Process** in your project- the most popular and influential iterative and incremental process framework.

Widely used in industry and incorporated into hybrid agile approaches such as Disciplined Agile Delivery (2012).

It is the most useful process framework to know

Unified Process (UP) structure

The UP organizes the product development spiral into 4 sequential phases. Each phase has its own distinct objectives and character:

Inception

Elaboration

Construction

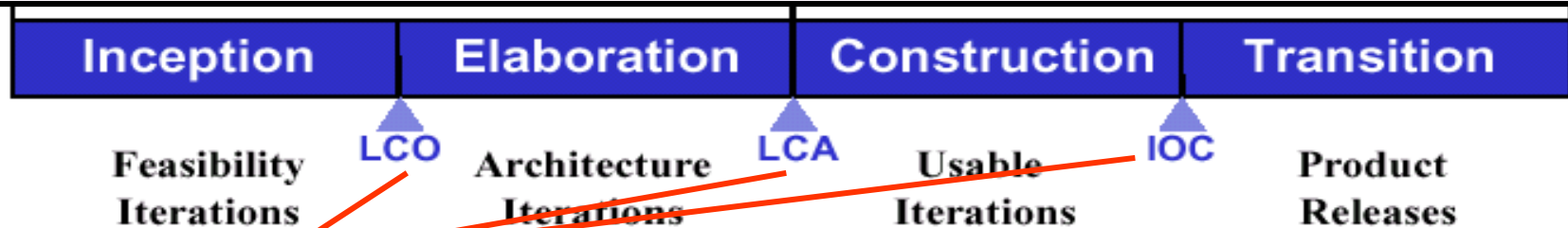
Transition

There are many ways to break projects into iterations within the UP framework, but all iterations in all phases are driven by use cases with a focus on risk reduction:

- use cases are specified and analysed,
- use cases are designed and implemented ("realized"),
- use cases are the source of test cases.

Phases of the Unified Process

No matter how you break into iterations, three anchor points are *UP invariants*



These "phase gates" are major milestones – commitment and progress checkpoints added to the spiral. They are called:

Life Cycle Objectives (LCO). Here you are ready to commit to develop an architecture.

Life Cycle Architecture (LCA). Here you are ready to commit to develop the product.

Initial Operational Capability (IOC). Here you are ready to commit to put it into use.

The phase gates are go/no-go decision points. They are "gates" because you can't pass them unless you have achieved the objectives of the phase.

If a project is going to fail, then you want to stop it as early as possible – this is another mechanism for reducing risk.

UP Phases – a simplified view that you’ve already seen!



Inception: Understand what to build

- Vision, high-level requirements, business case
- Addresses business risks

Elaboration: Understand how to build it

- *Verified* baseline architecture, most requirements detailed
- Addresses architectural/technical risks

Construction: Build the product

- Working product, system test complete
- Addresses logistical risks

Transition: Validate solution

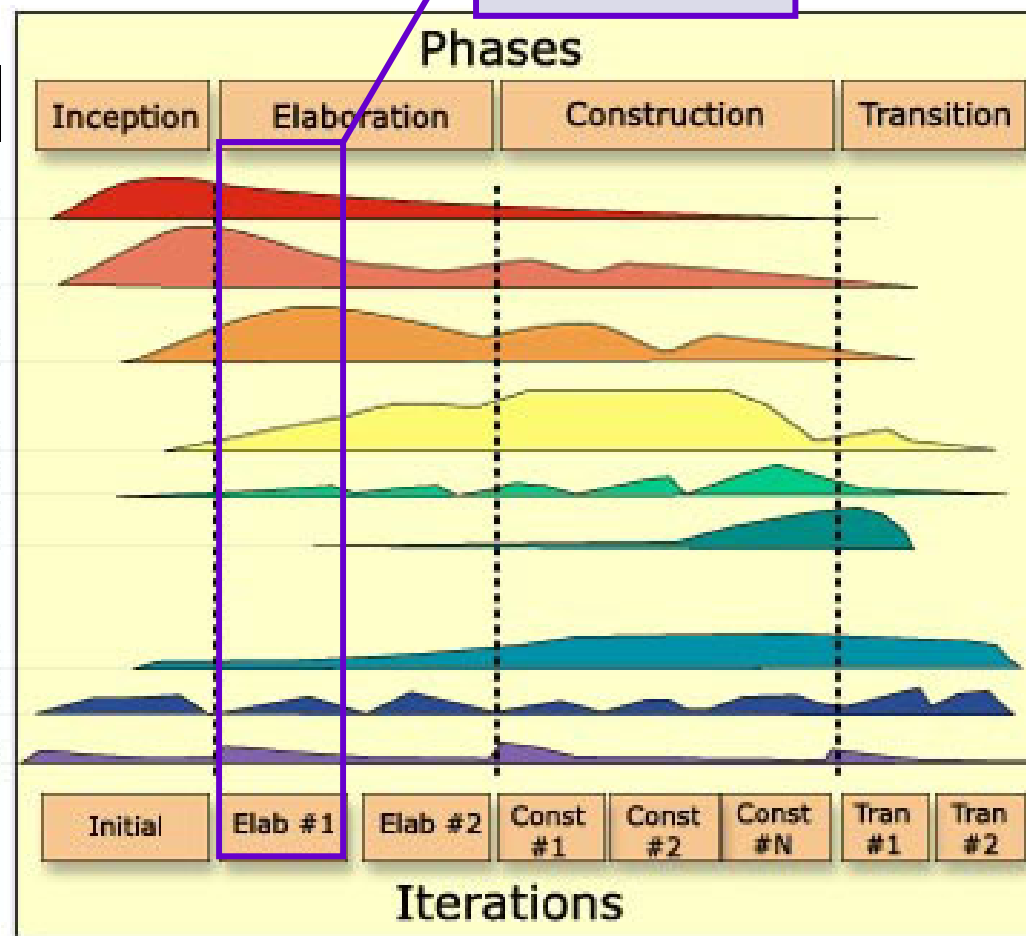
- Stakeholder acceptance
- Addresses delivery and roll-out risks

Unified Process framework

CONTENT
STRUCTURE

Disciplines

Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration
& Change Mgmt
Project Management
Environment



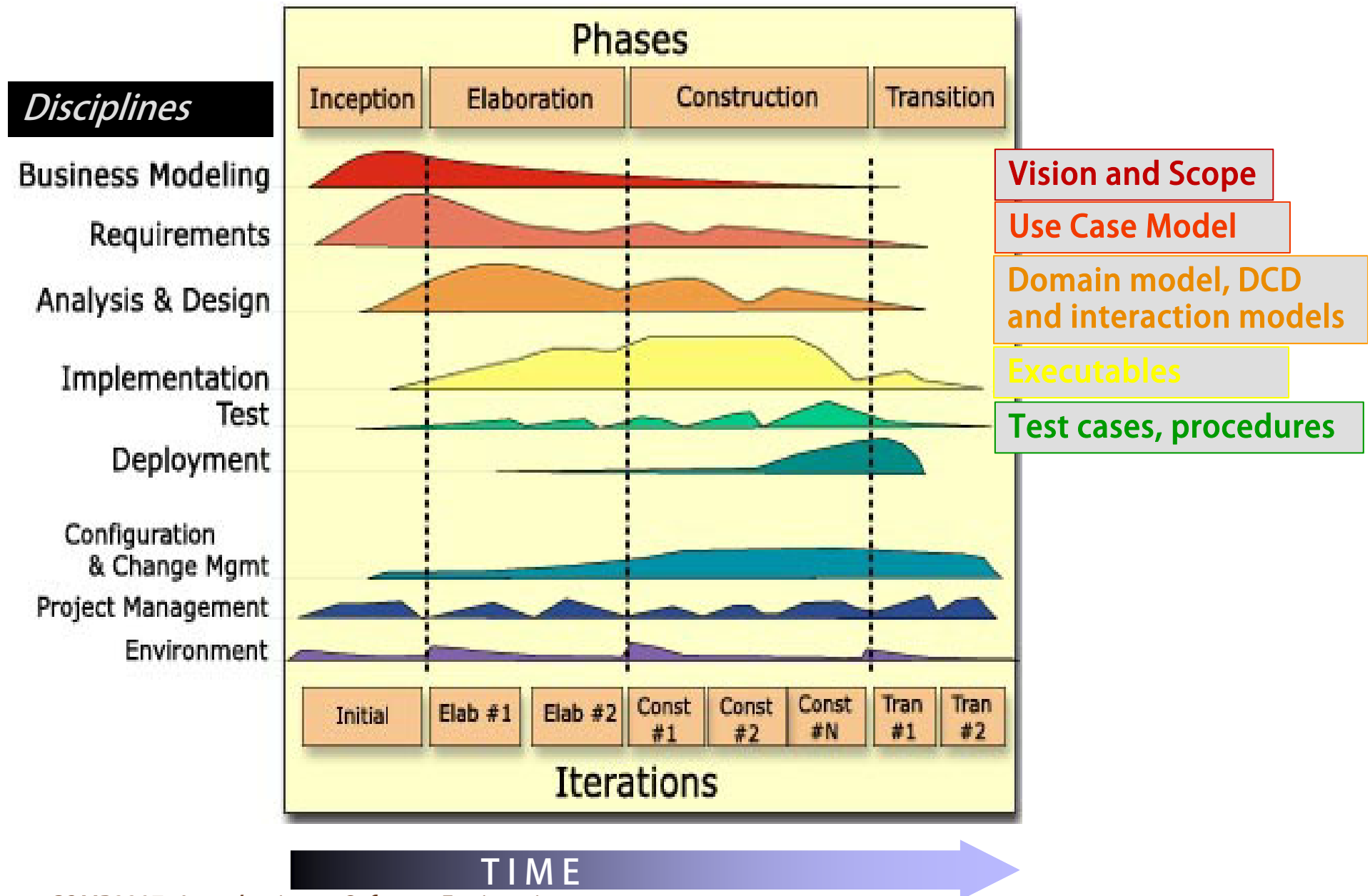
A distinct sequence of activities...

...with an established plan and evaluation criteria...

...resulting in an executable release.

If you don't understand this diagram, you don't understand the Unified Process!

Main artefacts of the Unified Process



Unified Process - Rough Planning



Three kinds of planning



A single, very coarse-grained *Project Plan*. The overall profile for the project. Big milestones on a timeline



Phase Plan. Iterations in a phase and their objectives. This is also at the level of detail that upper management is interested in.



Fine-grained *Iteration Plans*, one per iteration. Detailed planning within a *time-boxed* iteration.

"Time-boxed" means fixed length and slipping the end date is not allowed. If it seems the work of an iteration is taking too long, then remove tasks.

Handle them in a future iteration if still needed.

The Project Plan

Default Effort/Schedule Distributions

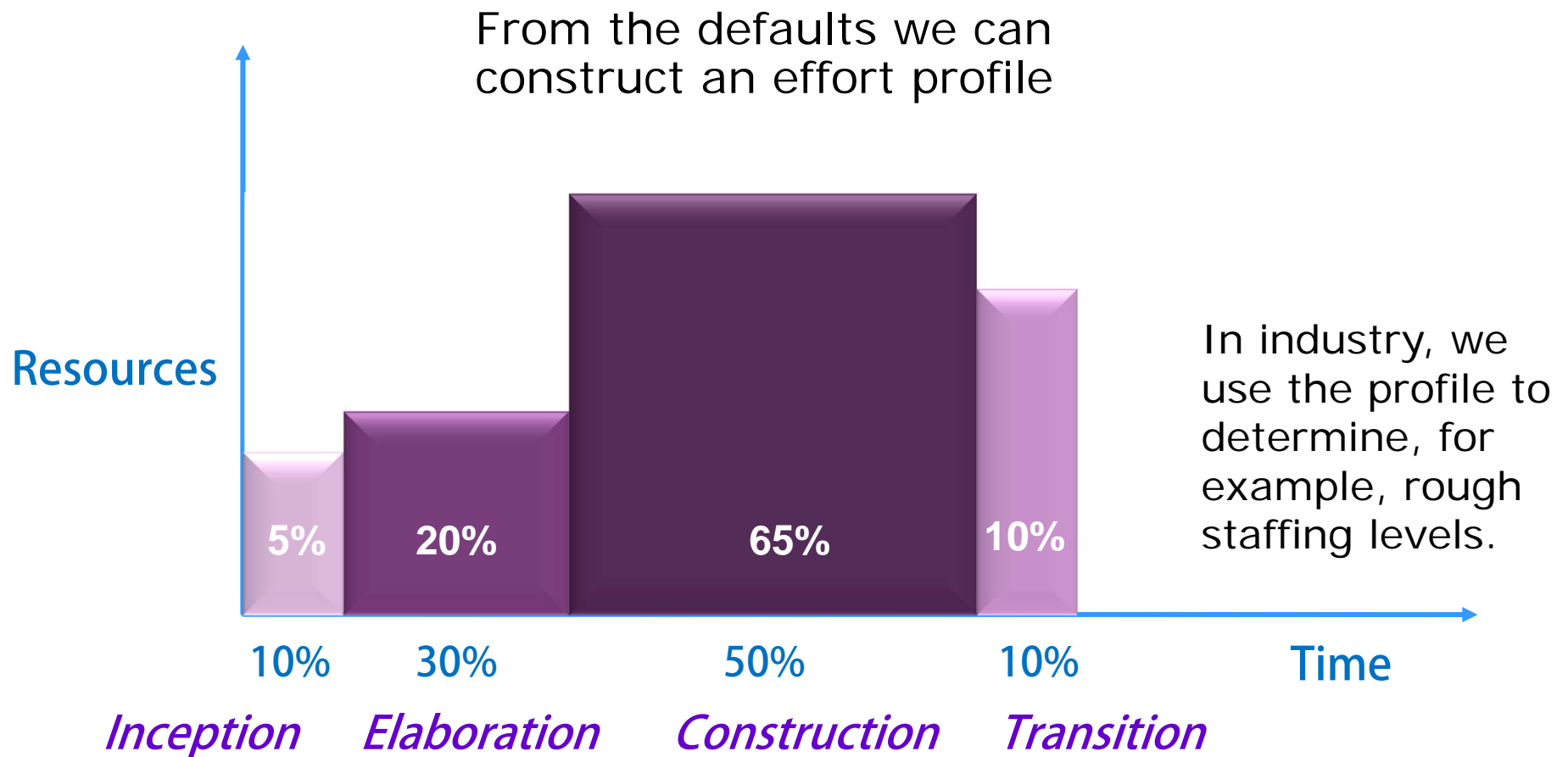
Begin with nominal profiles based on effort estimates and knowledge of typical project profiles.

- Medium size and effort.
- First development cycle of the product.
- No pre-existing architecture.
- Small number of risks and unknowns.

Rough defaults, extracted from *typical* projects:

	INCEPTION	ELABORATION	CONSTRUCTION	TRANSITION
Effort	5%	20%	65%	10%
Schedule	10%	30%	50%	10%

The Default Profile



Now we have an effort profile, we can begin to consider the nature of *this* particular project and modify the profile accordingly.

Duration of an Iteration - TimeBoxing in the UP

If too short: there won't be time to finish sufficient new work. Also, there is a management overhead for each iteration.

If too long: the complexity of the iteration increases. Also, feedback from users comes too late. It defeats the purpose of using UP

Around 2 to 6 weeks works well.

In industry, larger teams require longer because of the amount of coordination and communication needed.

In agile development with small teams, iterations are often shorter - 1 to 4 weeks.

Typical UP timeboxes

Project Length	Number of People	Duration of Iterations	Distribution
4 months	3	2-3 weeks	1, 1, 3, 1
8 months	10	4 weeks	1, 2, 3, 2
20 months	80	7-8	2, 3, 4, 2

Long iterations need internal milestones to manage them effectively. With big teams, each sub-system team can still do shorter “local” iterations by breaking each long iteration into several smaller ones.

Other approaches such as Scrum try to scale by increasing the number of teams rather than making teams bigger. 9 members is a common upper limit on team size.

Timeboxing in practice

- Gives a series of close, achievable deadlines
- Allows for tighter control, better decisions
- Good for team, good for client

Never slip the timebox deadline: do less.

Track progress and adjust scope early.

*The trick to making this work is to select the right stuff to postpone. These features are *not* automatically included in the next iteration.*

This keeps focus on the important goals and forces tradeoffs. Also prevents the project from grinding to a halt.

Short iterations need to be scoped very carefully.

Timeboxes don't have to be the same length throughout the project, but if they are it helps the team develop a "rhythm".

How Many Iterations in standard UP?

Inception

Sometimes 0 (or very short), particularly for a maintenance release.

More (1-2) if:

- You need to develop requirements/build UI prototype
- You need to demonstrate a proof of concept
- You need to make the business case and find funding

Elaboration

These are hard

Minimum of 1 if building on a well-established architectural framework.

If not, then 2 is better to achieve a good architectural baseline (1 for the architectural prototype, 1 for the architectural baseline).

3 if there are many risks or new factors.

Standard UP iterations (cont.)

These are expensive

Construction

At least 1 leading to the beta release.

Usually one more for a partially-complete system before beta.

Often good reasons to add a third for larger projects. There is a lot of work in Construction

Transition

Usually 1 for the transition from beta release to final product.

Possibly 2, to allow for more feedback and rework.

Thus:

A total of between 3 and 10 iterations

6-8 is the most common range

Rough Rule for UP executables for Normal Projects

A typical mid-size project will deliver 6 or more executables:

One exploratory prototype in Inception

Two in Elaboration – an architectural prototype and a baseline

Two in Construction – alpha and beta releases

One in transition – the final product release

More agile projects tend to use shorter iterations and more of them.