

40431: Modelação e Análise de Sistemas

O processo de desenvolvimento de sistemas (de software)

Ilídio Oliveira

v2020-10-15

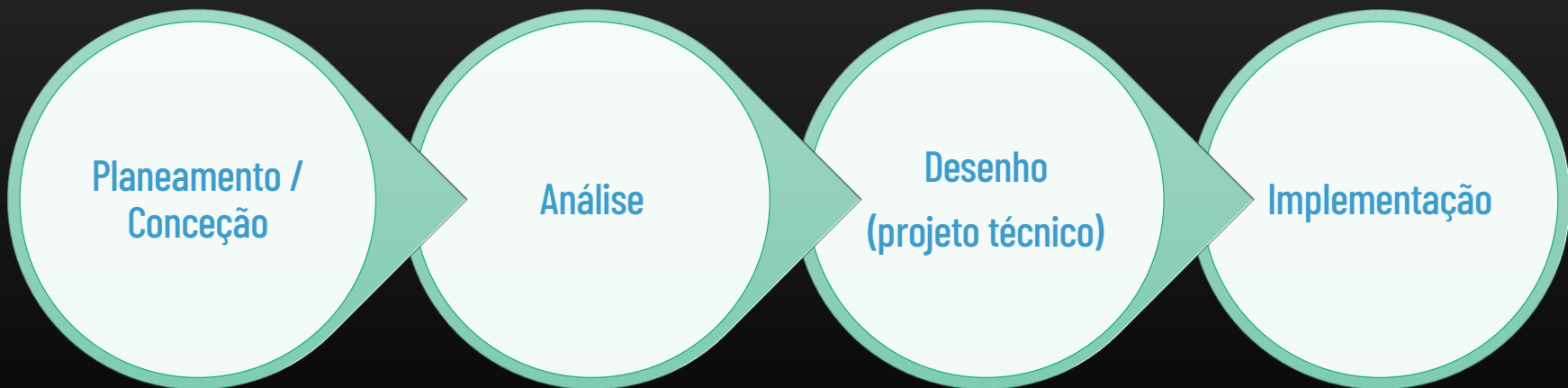
Objetivos de aprendizagem

- Identificar atividades comuns a todos os projetos (ciclo de vida)
- Distinguir projetos sequenciais de projetos evolutivos
- Descrever a estrutura do Unified Process (fases, objetivos, iterações)
- Identificar as principais atividades exigidas na atribuição do projeto
- Mapear disciplinas técnicas nas fases do OpenUP

Software Development lifecycle

Quatro fases fundamentais: planejamento/conceito, análise, desenho e implementação. Diferentes projetos podem enfatizar diferentes partes do SDLC ou realizar as fases SDLC de diferentes formas, mas todos os projetos têm elementos destas quatro fases.

Cada fase é composta por uma série de atividades, em que aplica disciplinas técnicas para produzir resultados previstos



Fases fundamentais: planeamento, análise, desenho e implementação

A fase de planeamento é o processo fundamental de compreensão do porquê de um sistema de informação ser construído e determinar como a equipa do projeto irá construí-lo.

Atividades-chave:

1. Arranque do projeto ("OK")

O valor que o sistema gerará para a organização é identificado. O pedido do sistema e a análise de viabilidade são apresentados a um comité de aprovação que decide se o projeto deve ser realizado.

2. Gestão do projeto

O gestor do projeto cria um plano de trabalho, equipa o projeto, e escolhe técnicas para a equipa controlar e dirigir o projeto através de todo o SDLC.

Fases fundamentais: planejamento, análise, desenho e implementação

A fase de análise responde às questões de quem irá usar o sistema, o que o sistema vai fazer, e onde e quando será utilizado.

Durante esta fase, a equipa do projeto investiga qualquer sistema atual, identifica oportunidades de melhoria e desenvolve um conceito para o novo sistema.

Atividades-chave:

1. Análise dos sistemas existentes,
2. Recolha de requisitos (necessidades da organização)
3. Conceito de solução (proposta do sistema)

Fases fundamentais: planejamento, análise, desenho e implementação

A fase de desenho decide como o sistema funcionará, em termos de hardware, software e infraestrutura de rede; a interface, formulários e relatórios do utilizador; e os programas específicos, bases de dados e ficheiros que serão necessários.

Atividades-chave:

1. Estratégia de desenvolvimento (interna ou contratual?)
2. Desenho da arquitetura do sistema
3. Desenho do modelo de dados
4. Desenho dos programas (classes, etc.)

Fases fundamentais: planejamento, análise, desenho e implementação

Na fase de implementação, o sistema é realmente construído

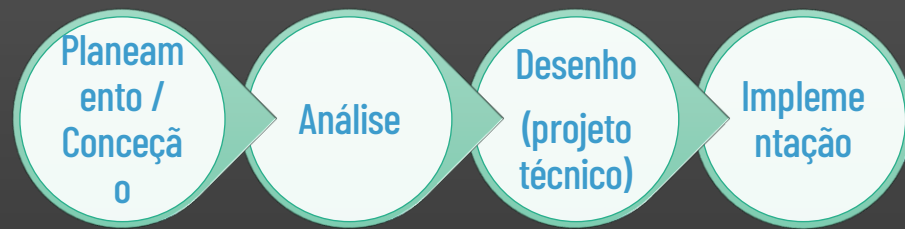
(ou adquirido, no caso de integração de pacotes existentes).

Inclui também a transição para o ambiente de produção.

Atividades-chave:

1. Implementação de sistemas (construção e garantia de qualidade)
2. Instalação e transição
3. Plano de suporte (revisão pós-instalação e gestão de alterações)

Para quê um "processo"?



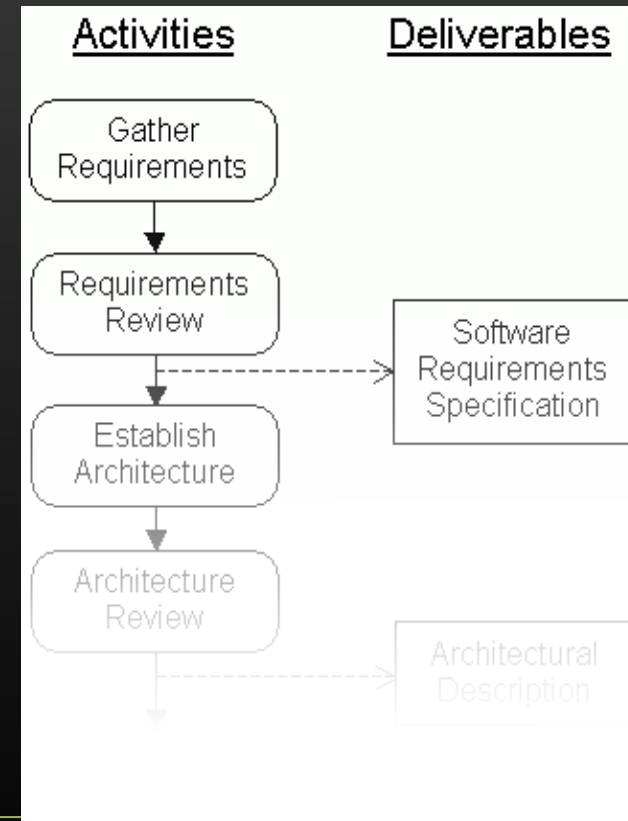
SDLC (visão geral do desenvolvimento)

O SDLC é concretizado usando um processo de software sistemático.

Por que precisamos de um processo formal?

- Falhas ocorrem (muito) frequentemente
- Criar sistemas não é intuitivo
- Os projetos são terminados fora do prazo, acima do orçamento ou entregues com menos funcionalidades do que o previsto

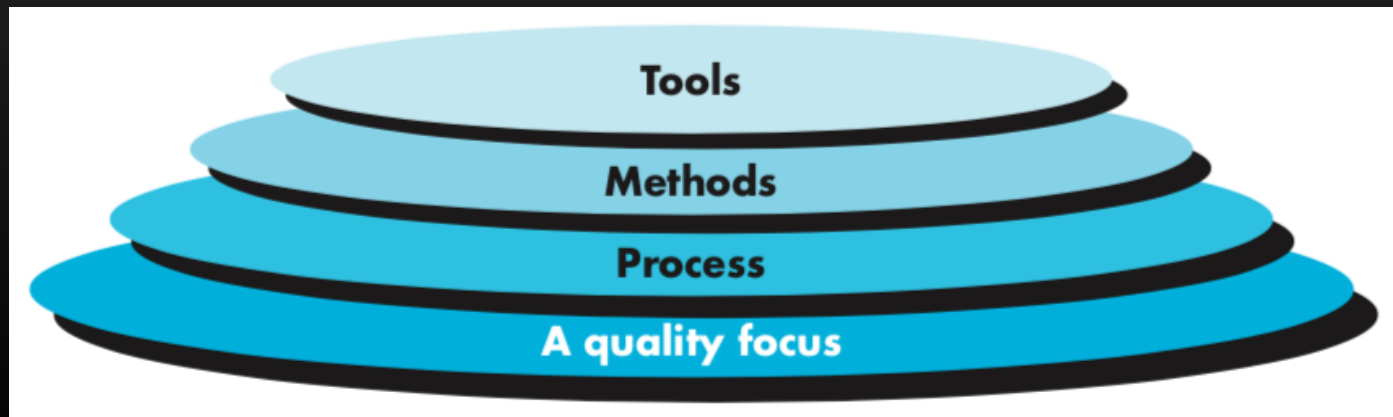
Processo ("guião")



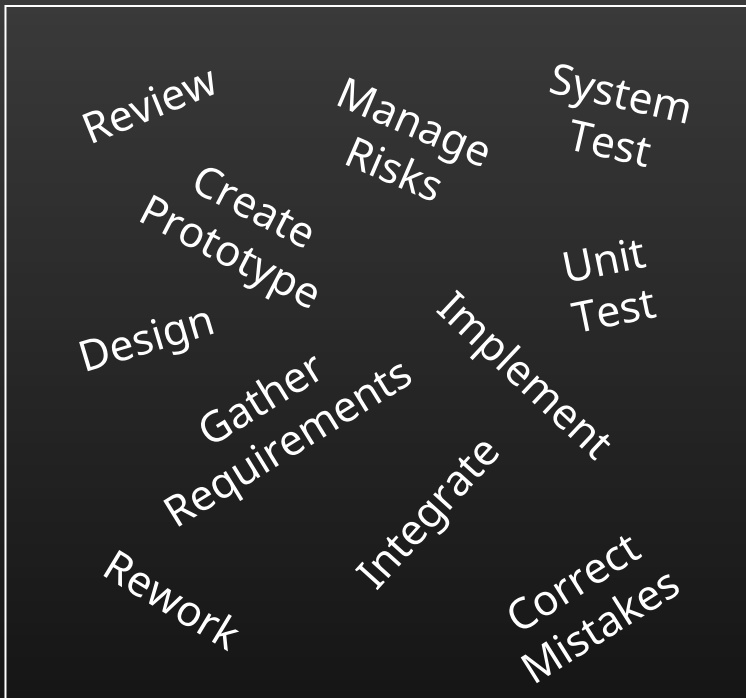
Processo / *software process*

Um processo de software é um guião para as atividades, ações e tarefas que são necessárias para construir software de alta qualidade.

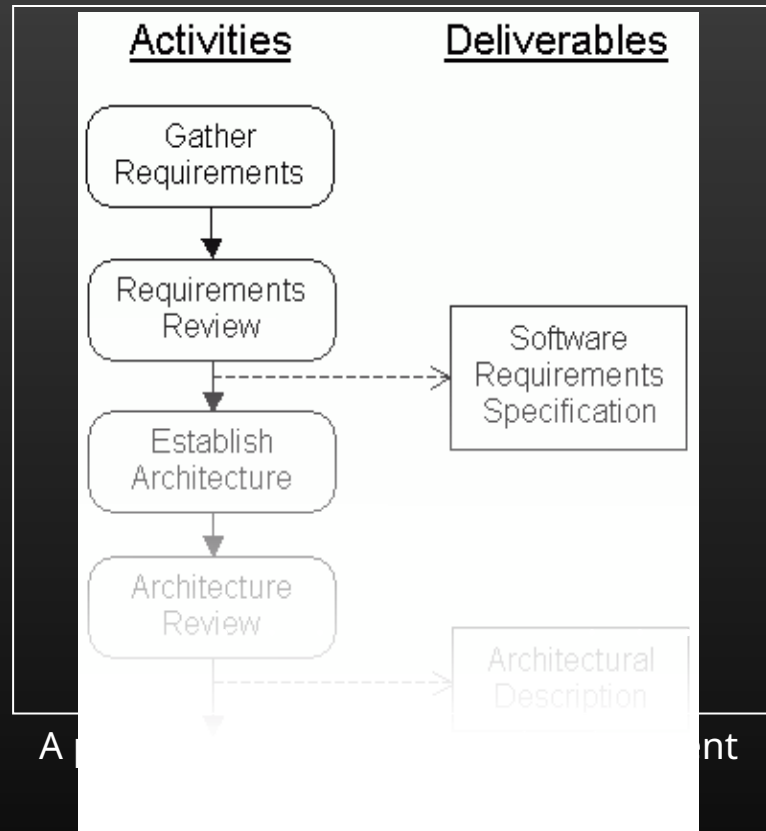
Estabelece o quadro técnico e de gestão para a aplicação de métodos, ferramentas e papéis (de pessoas) no desenvolvimento.



Why Software Process?



Developing software without a defined process is chaotic and inefficient



"It is better not to proceed at all, than to proceed without method." -- Descartes

Software process

A process specifies:

What?

Who?

How?

When?

A process includes:

Roles

Workflows

Procedures

Standards

Templates

There is no single “best process”

Organizations should select (or customize) their process.

http://sweet.ua.pt/ico/OpenUp/OpenUP_v1514/

Software process description

When we describe and discuss processes, we usually talk about ...

the **activities** in these processes such as specifying a data model, designing a user interface, etc. and the **sequence** of these activities.

Process descriptions may also include:

Products, which are the outcomes of a process activity;

Roles, which reflect the responsibilities of the people involved in the process;

Pre- and post-conditions (dependencies), which are statements that are true before and after a process activity has been enacted or a product produced.

Categories for software development methodologies

Structured Analysis and Design

Structured to replace previous “ad-hoc”

formal step-by-step approach to the SDLC that moves logically from one phase to the next

e.g.: Waterfall

Rapid-Application Development

Get some part of the system developed quickly and into the hands of the user (early feedback and value)

e.g.: Phased/Incremental, Prototyping

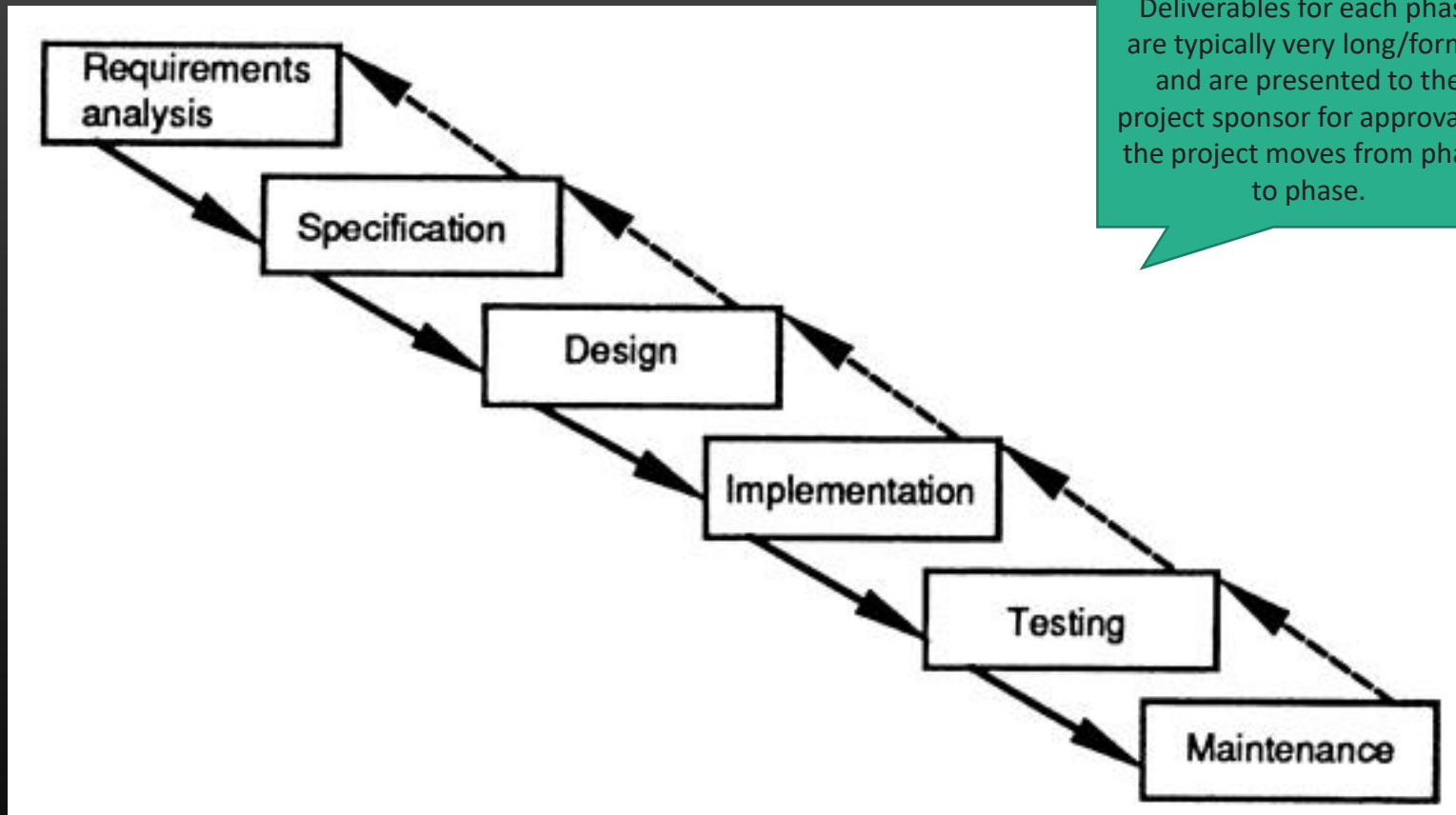
Agile development

focus on eliminating much of the modeling and documentation overhead; emphasize simple, iterative application development

A “mindset” to the development

e.g.: Extreme Programming

“Classical” engineering approach: Waterfall model



W. Royce, “Managing the Development of Large Software Systems,” *Proc. Westcon*, IEEE CS Press, 1970, pp. 328-339.

Waterfall model advantages

Simple and easy to understand and use.

Easy to plan

A schedule can be set with deadlines for each stage of development and a product can proceed through the development process like a car in a car-wash, and theoretically, be delivered on time.

Easy to manage

each phase has specific deliverables and a review process.

Phases are processed and completed one at a time.

Works well where requirements are stable and well understood

Waterfall model disadvantages

Problems

Difficulty of accommodating change after the process is underway.

Poor model for long and ongoing projects.

No working software is produced until late during the life cycle.

Not suitable for the projects where requirements are uncertain or at the risk of changing.

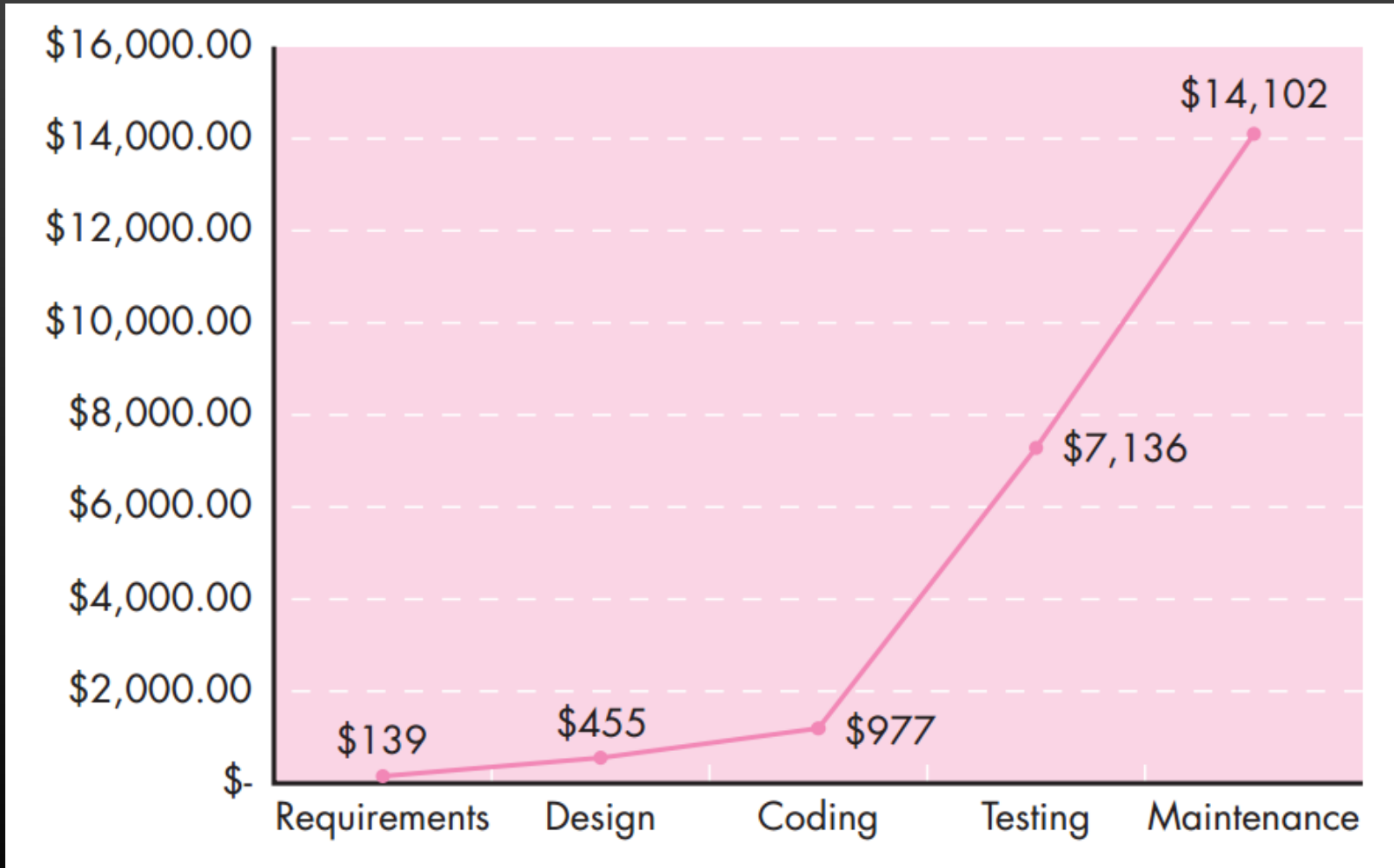
Why it may fail?

Real projects rarely follow the sequential flow that the model proposes

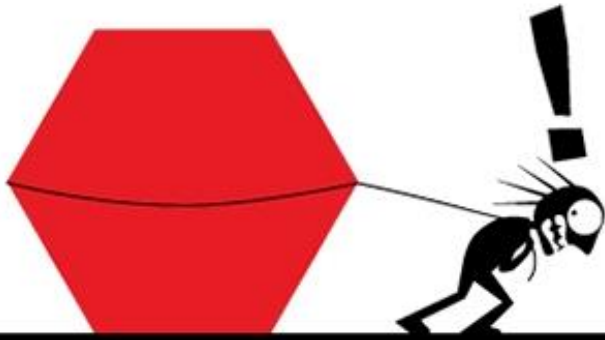
It is often difficult for the customer to state all requirements explicitly

The customer must have patience. A working version of the program(s) will not be available until late in the project time span

The cost of correcting an error raises exponentially along the sw lifecycle



One project? Micro-projects?



*'This project has got so big,
I'm not sure I'll be able to deliver it!'*



*'It's so much better delivering this
project in bite-sized sections'*

<https://blog.ganttpro.com/en/waterfall-vs-agile-with-advantages-and-disadvantages/>

Phased / incremental

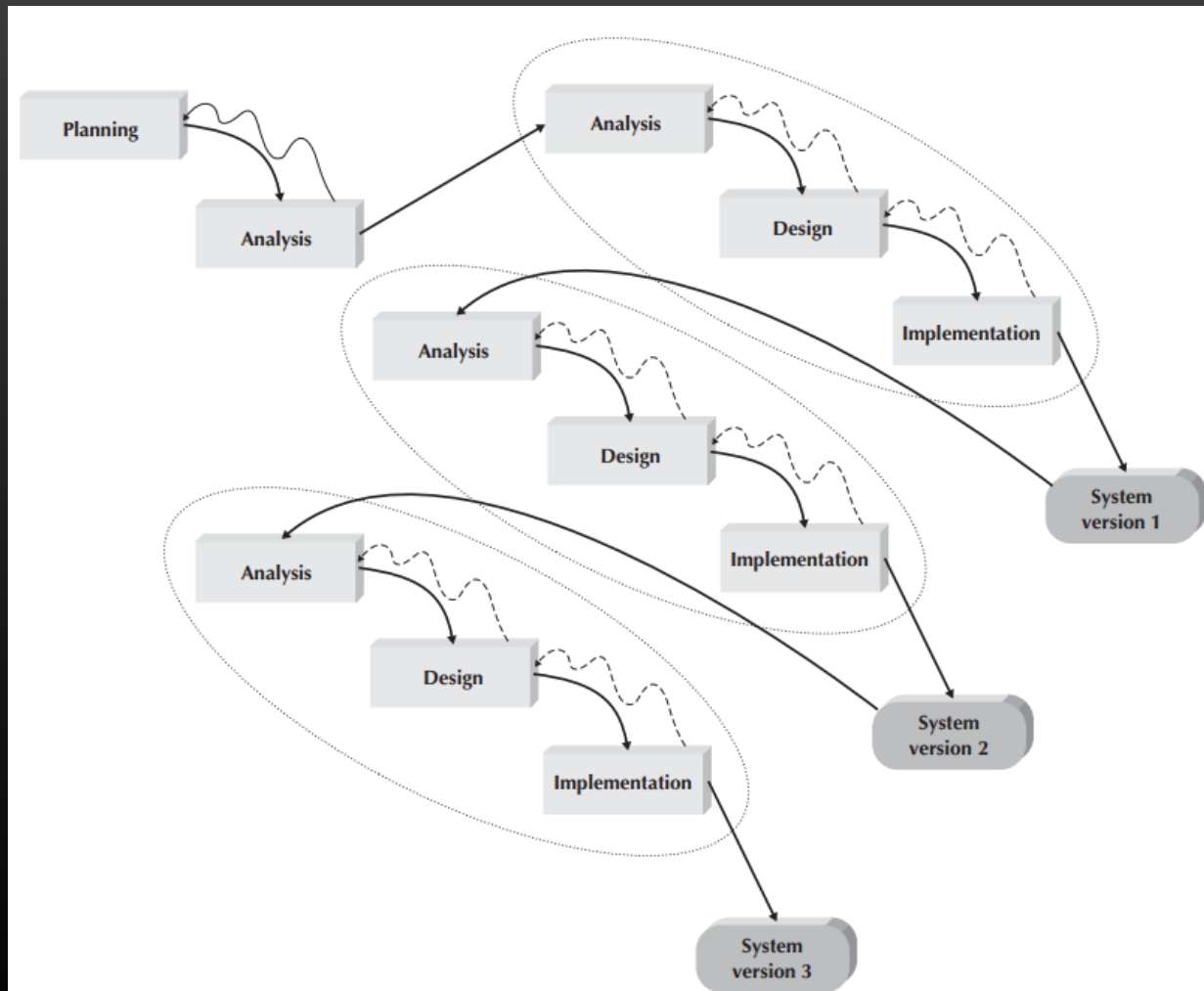


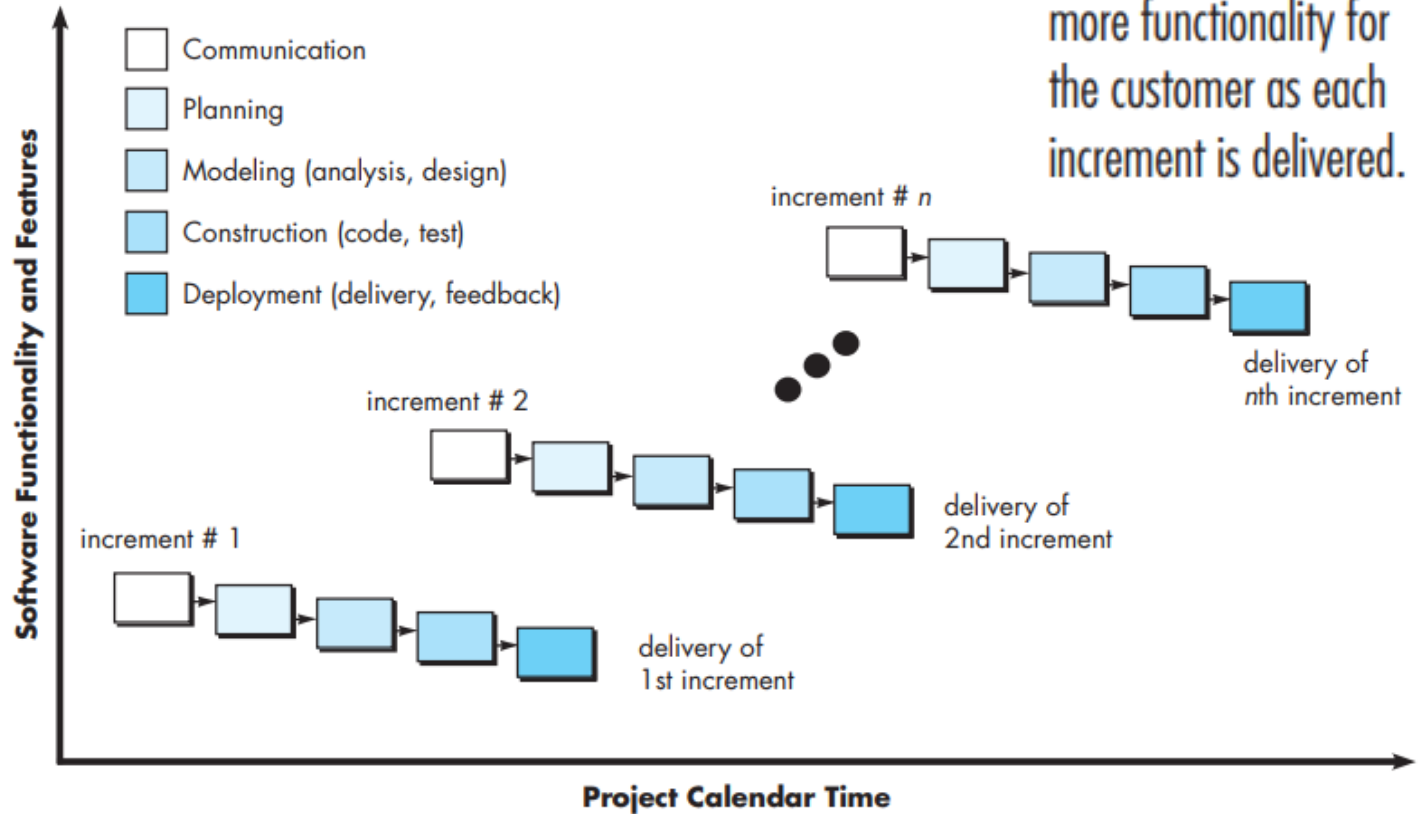
FIGURE 1-4 A Phased Development-Based Methodology

RAD: phased/incremental

The incremental model delivers a series of releases, called increments, that provide progressively more functionality for the customer as each increment is delivered.

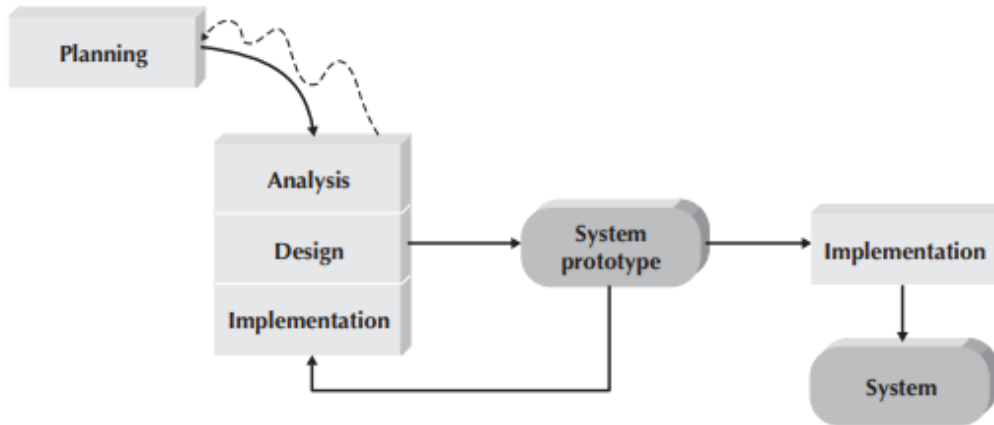
FIGURE 4.3

The incremental model



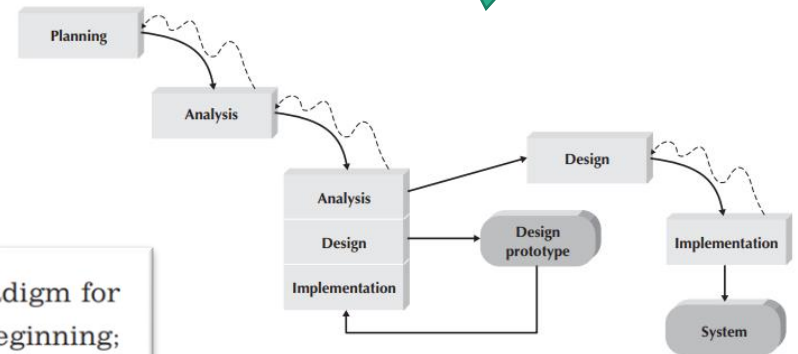
RAD: prototyping

FIGURE 1-5
A Prototyping-Based Methodology



Throwaway prototyping: a variation of Prototyping in which the prototype is discarded (not used for the implementation).

Although problems can occur, prototyping can be an effective paradigm for software engineering. The key is to define the rules of the game at the beginning; that is, all stakeholders should agree that the prototype is built to serve as a mechanism for defining requirements. It is then discarded (at least in part), and the actual software is engineered with an eye toward quality.



A Throwaway Prototyping-Based Methodology

Plan-driven and agile processes

Plan-driven/conventional processes

all of the process activities are planned in advance and progress is measured against this plan.

Agile processes

planning is incremental and it is easier to incorporate changes to reflect changing customer requirements.



<http://10aw.jf.xsl.pt/>



What is “Agility” in software development?

Effective (rapid and adaptive) response to change

Effective communication among all stakeholders

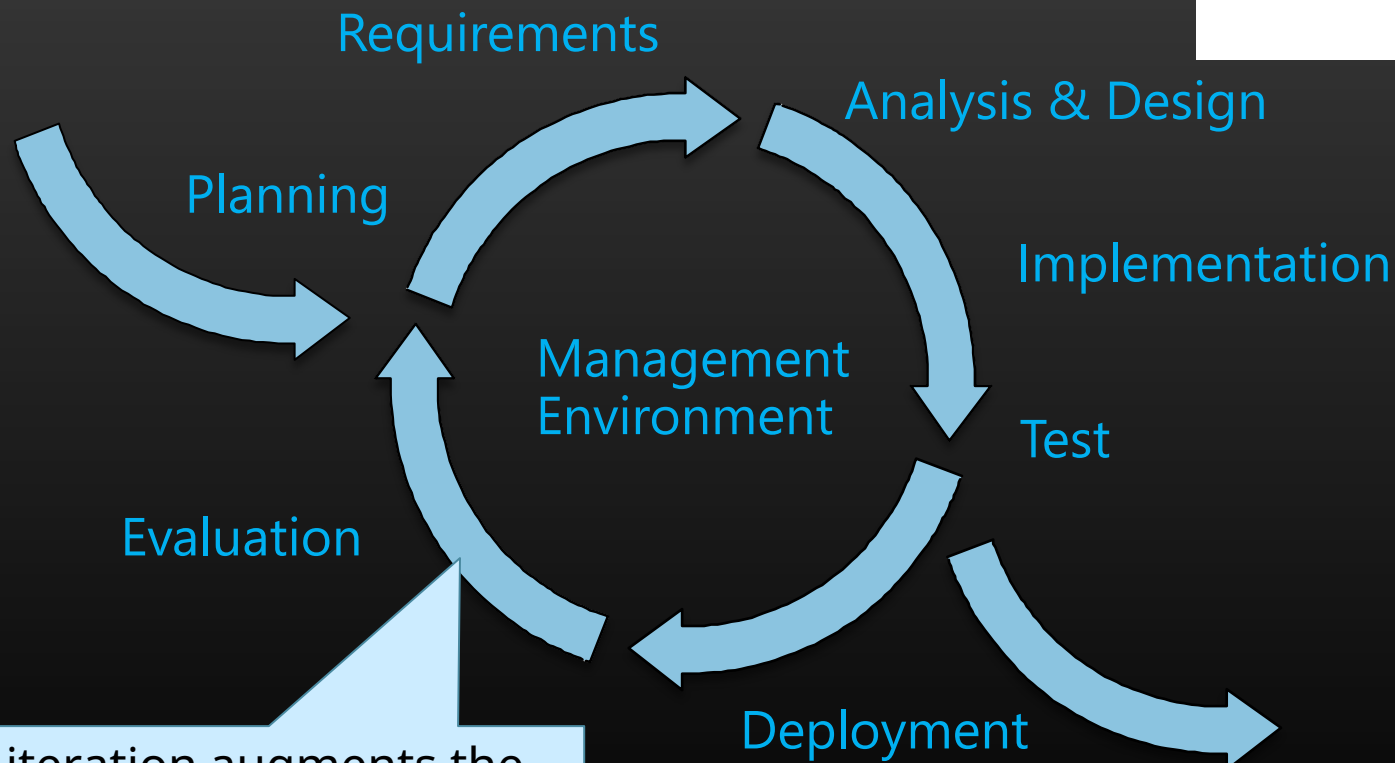
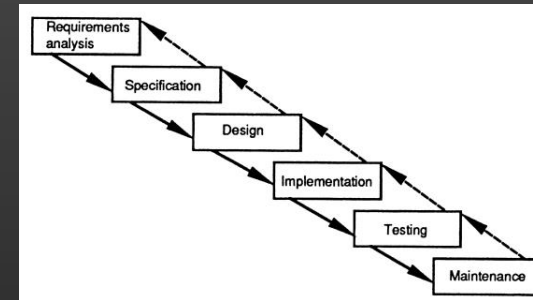
Drawing the customer onto the team

Organizing a team so that it is in control of the work performed

Yielding ...

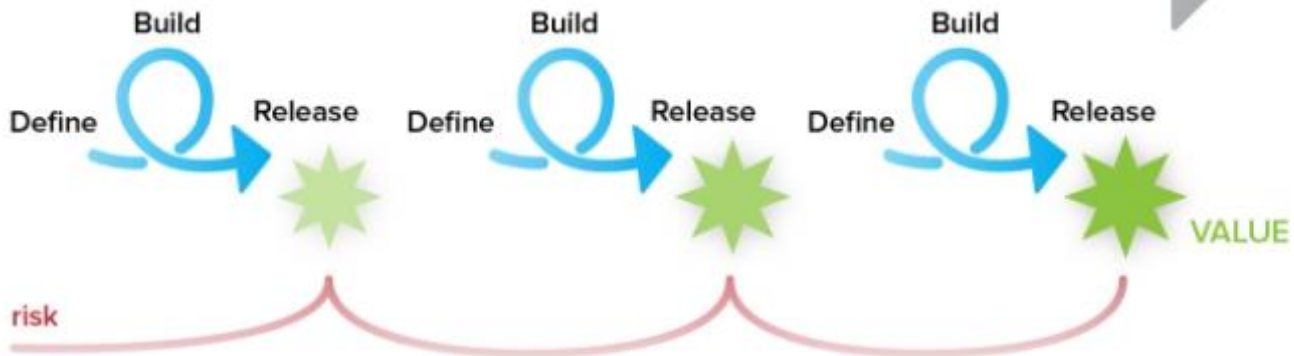
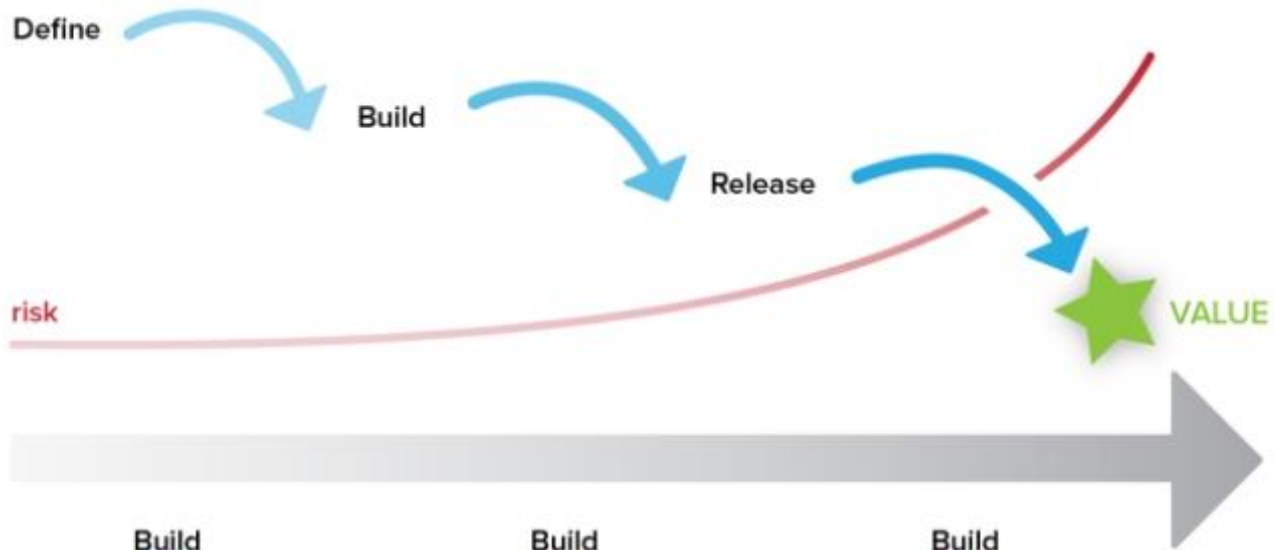
Rapid, incremental delivery of software

Agile is incremental and iterative development focuses on short and value-oriented cycles



Each iteration augments the solution by integrating some executable result

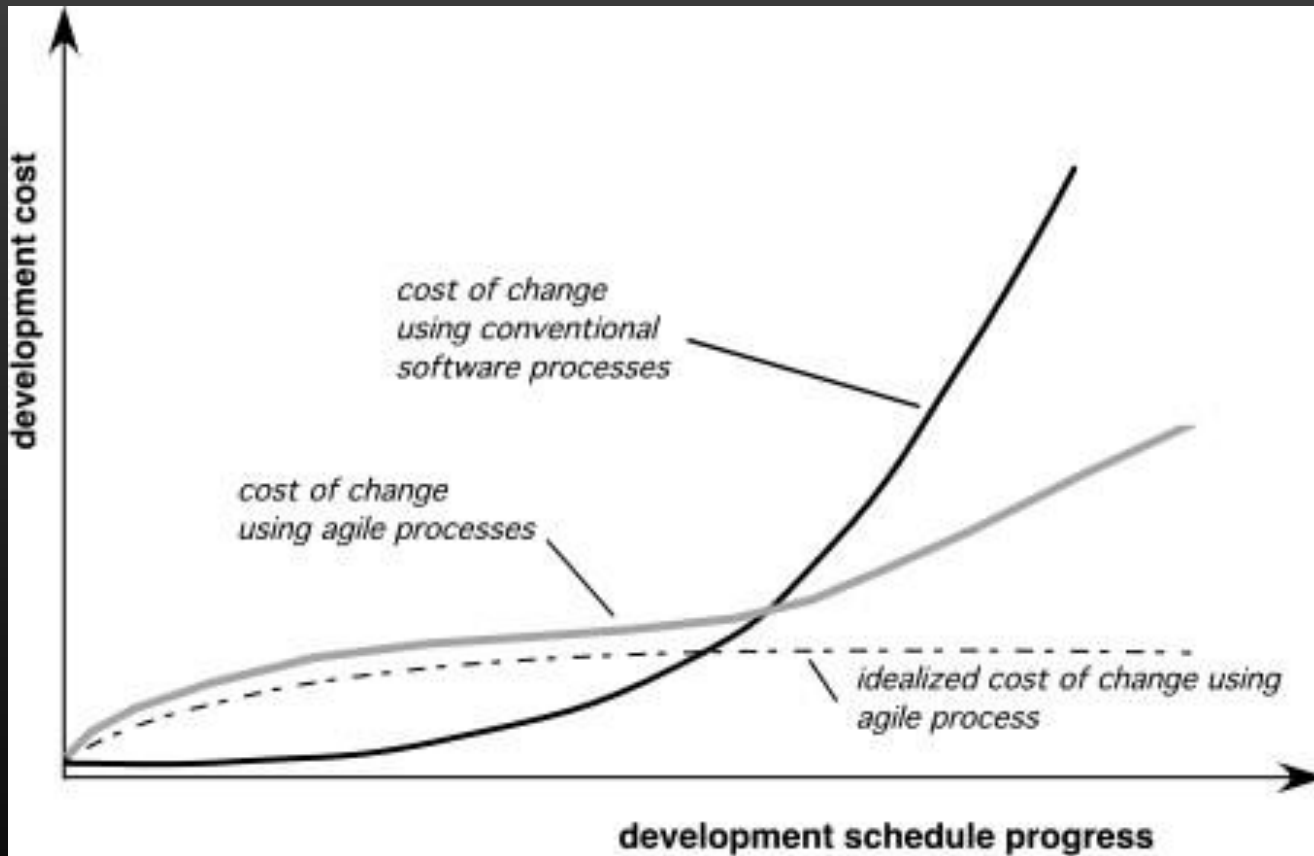
WATERFALL



AGILE

<https://blog.ganttpro.com/en/waterfall-vs-agile-with-advantages-and-disadvantages/>

Agility and the Cost of Change



The cost of change increases nonlinearly as the project progresses

To be (agile) or not to be...

WATERFALL



AGILE



Iterative development (short cycles) vs linear development through stages?

Frequent business interaction vs fluctuations in stakeholder's participation?

Best to have good collaboration or a good plan?

Welcome changes to mitigate risk vs avoid changes to control risk?

Teams can select from several methodologies

Ability to Develop Systems	Structured Methodologies		RAD Methodologies			Agile Methodologies	
	Waterfall	Parallel	Phased	Prototyping	Throwaway Prototyping	XP	SCRUM
With Unclear User Requirements	Poor	Poor	Good	Excellent	Excellent	Excellent	Excellent
With Unfamiliar Technology	Poor	Poor	Good	Poor	Excellent	Good	Good
That Are Complex	Good	Good	Good	Poor	Excellent	Good	Good
That Are Reliable	Good	Good	Good	Poor	Excellent	Excellent	Excellent
With a Short Time Schedule	Poor	Good	Excellent	Excellent	Good	Excellent	Excellent
With Schedule Visibility	Poor	Poor	Excellent	Excellent	Good	Excellent	Excellent

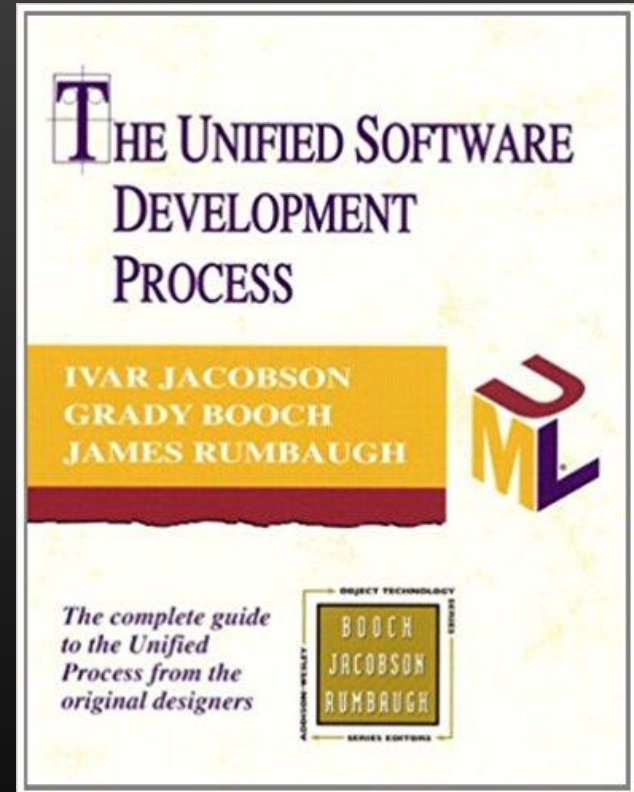
FIGURE 1-8 Criteria for Selecting a Methodology

Unified Process/Open Unified Process

An attempt for a general-purpose process

Can be adapted to the specific project in hands

OpenUP is a “free” version of the Unified Process



THE UNIFIED SOFTWARE DEVELOPMENT PROCESS

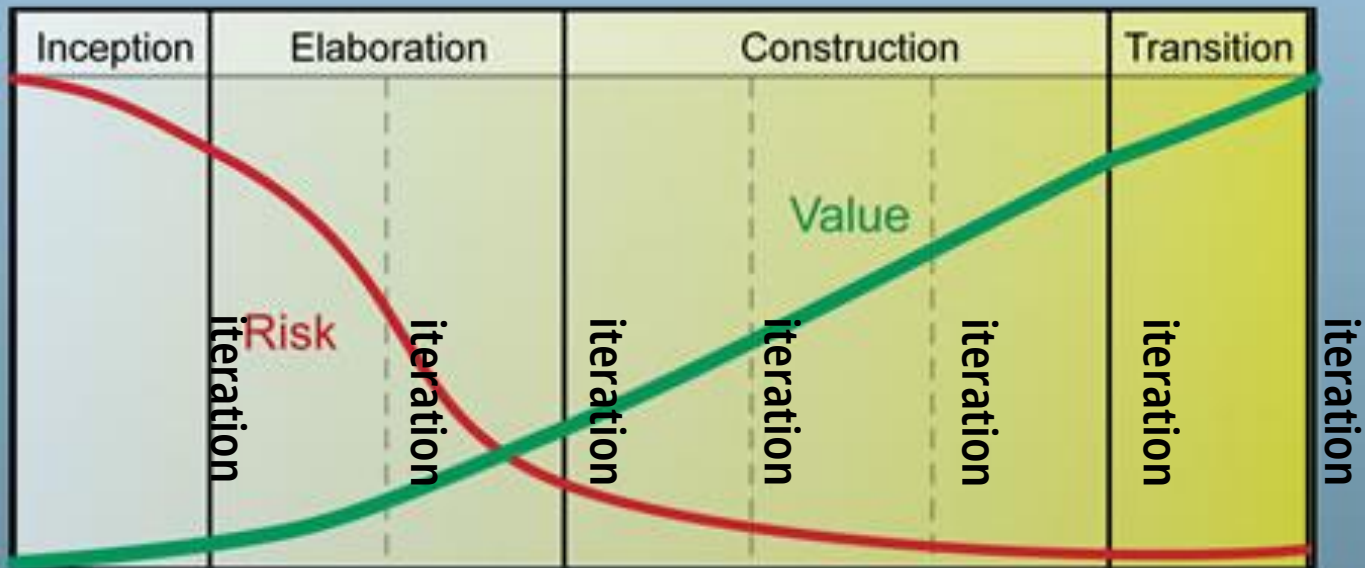
IVAR JACOBSON
GRADY BOOCH
JAMES RUMBAUGH



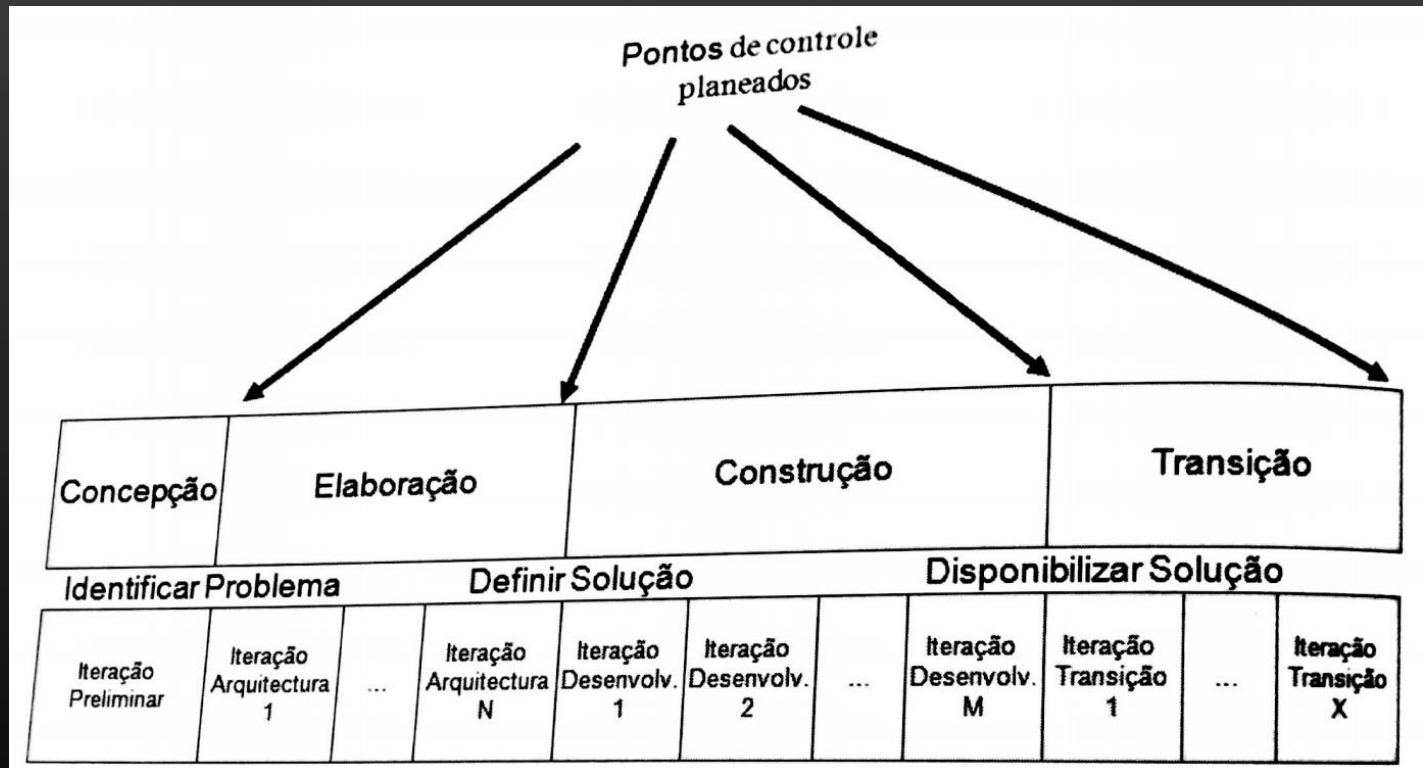
*The complete guide
to the Unified
Process from the
original designers*



Project Lifecycle



PT: Fases, iterações e pontos de controlo



OpenUP/Unified Process activities

The UP offers an approach to the SDLC visualized as a **matrix**, crossing different **technical disciplines** with evolving **iterations** in the project. (Note: UP phases \neq SDLC phases)

Requirements analysis is mainly performed at the beginning of the project (requirements baseline) but also during the iterations (evolutionary requirements).

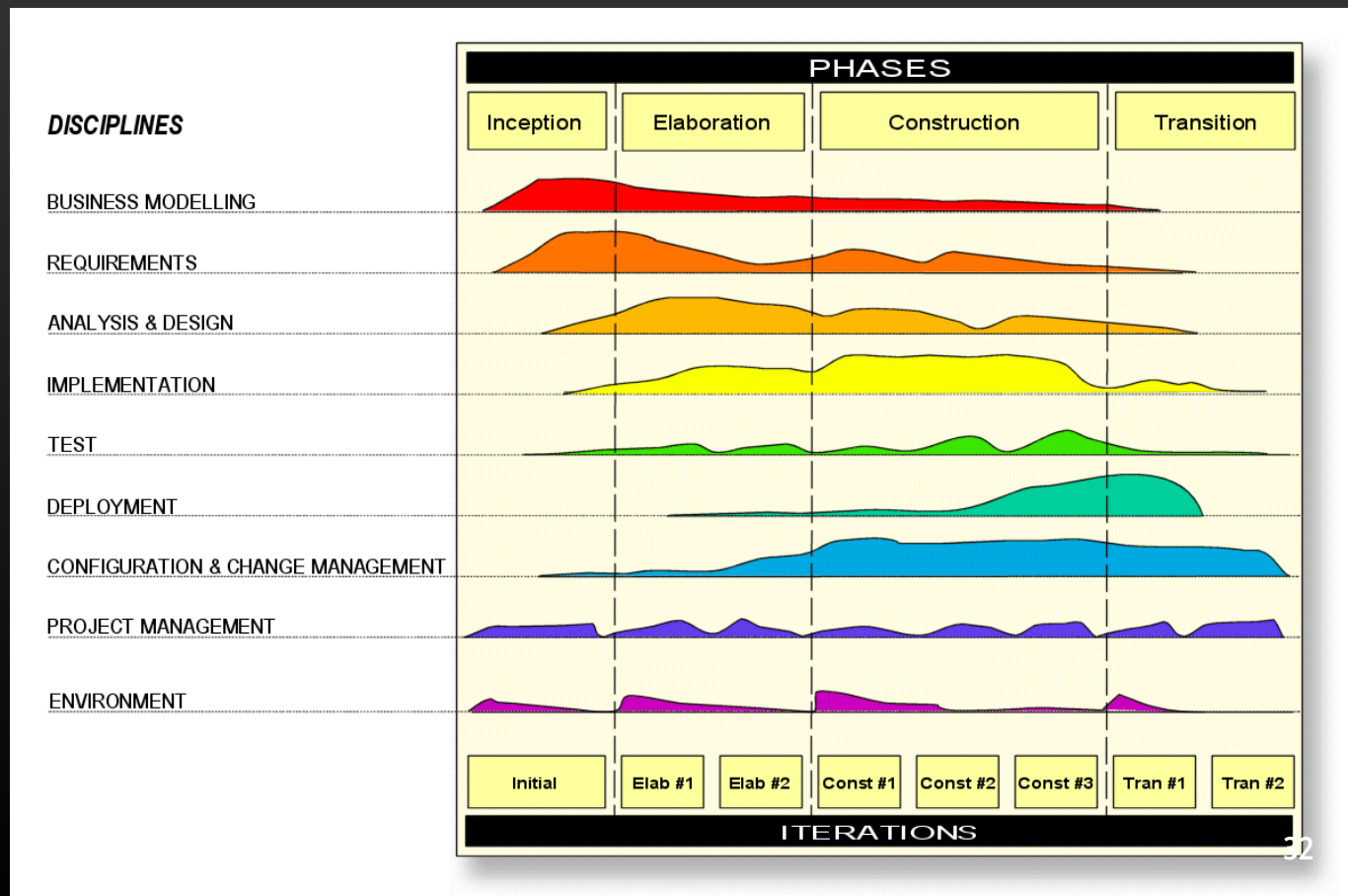
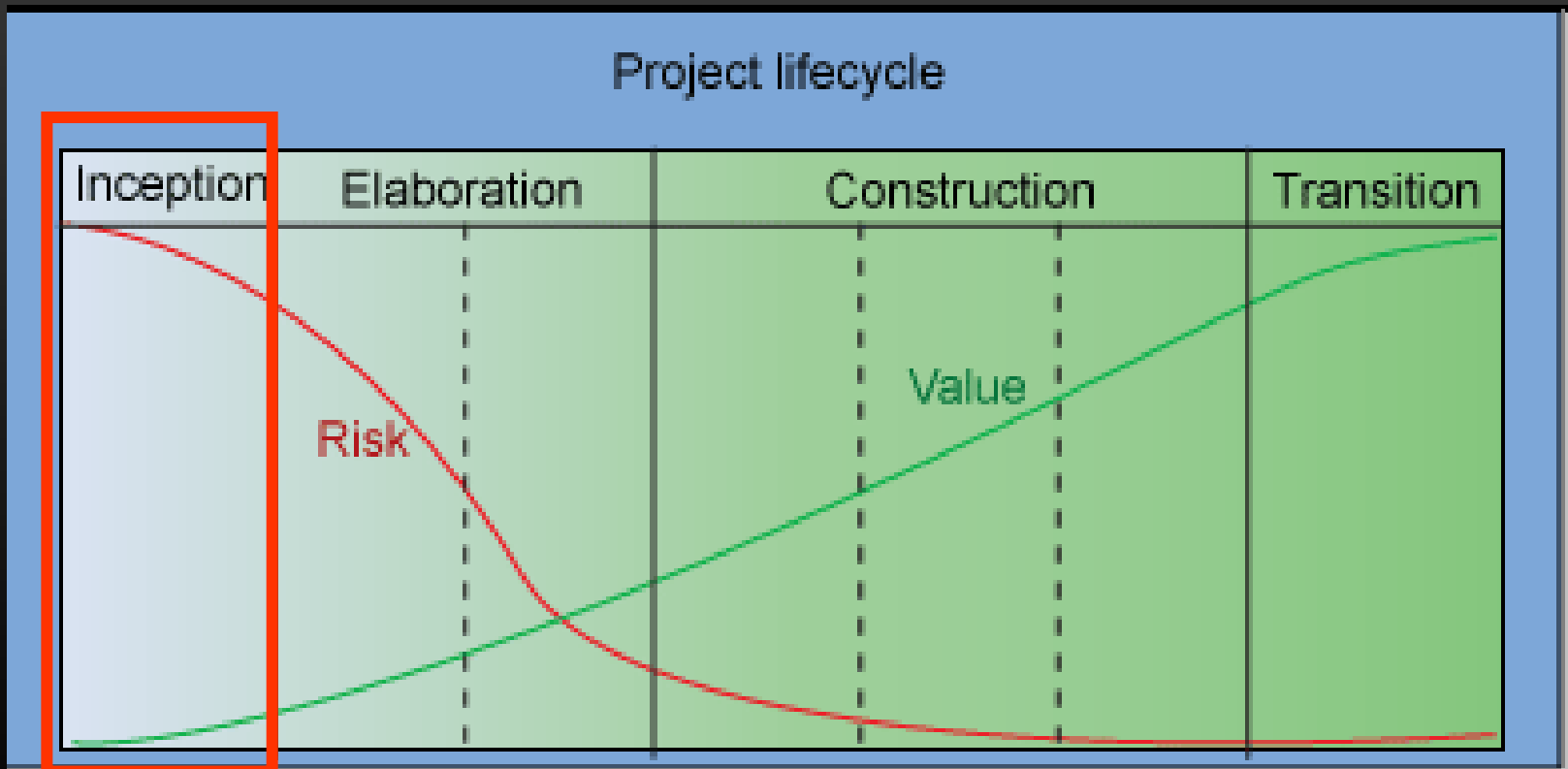


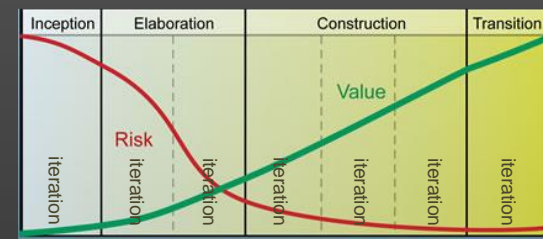
Figura Project lifecycle

The phases: Inception

Do we agree on project scope and objectives, and whether or not the project should proceed?



Inception: Know What to Build



Typically one short iteration

Produce vision document and initial business case

Develop high-level project requirements

Initial use-case and (optional) domain models (10-20% complete)

Focus on what is required to get agreement on 'big picture'

Manage project scope

Reduce risk by identifying key requirements

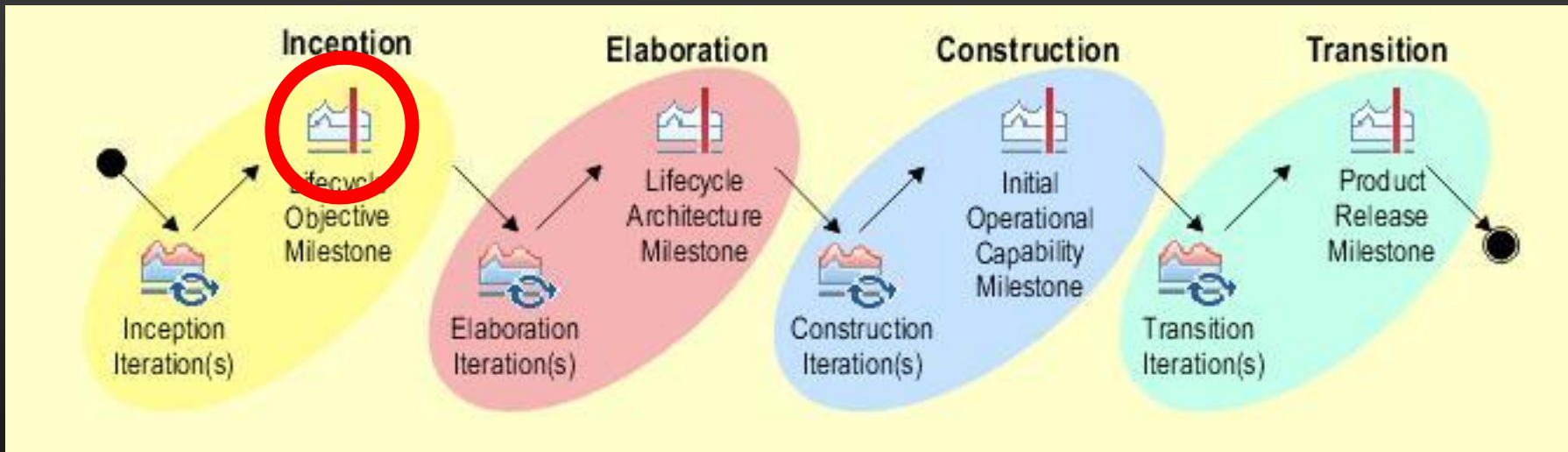
Acknowledge that requirements will change

Manage change, use iterative process

Produce conceptual prototypes as needed

Credit: Per Kroll (IBM)

Milestone: Inception



Lifecycle Objectives Milestone. At this point, you examine the cost versus benefits of the project, and decide either to proceed with the project or to cancel it.

Elaboration: Know How to Build It by Building Some

Elaboration can be a day long or several iterations

Balance

mitigating key technical and business risks with producing value (tested code)

Produce (and validate) an executable architecture

Define, implement and test interfaces of major components.
Partially implement some key components.

Identify dependencies on external components and systems.
Integrate shells/proxies of them.

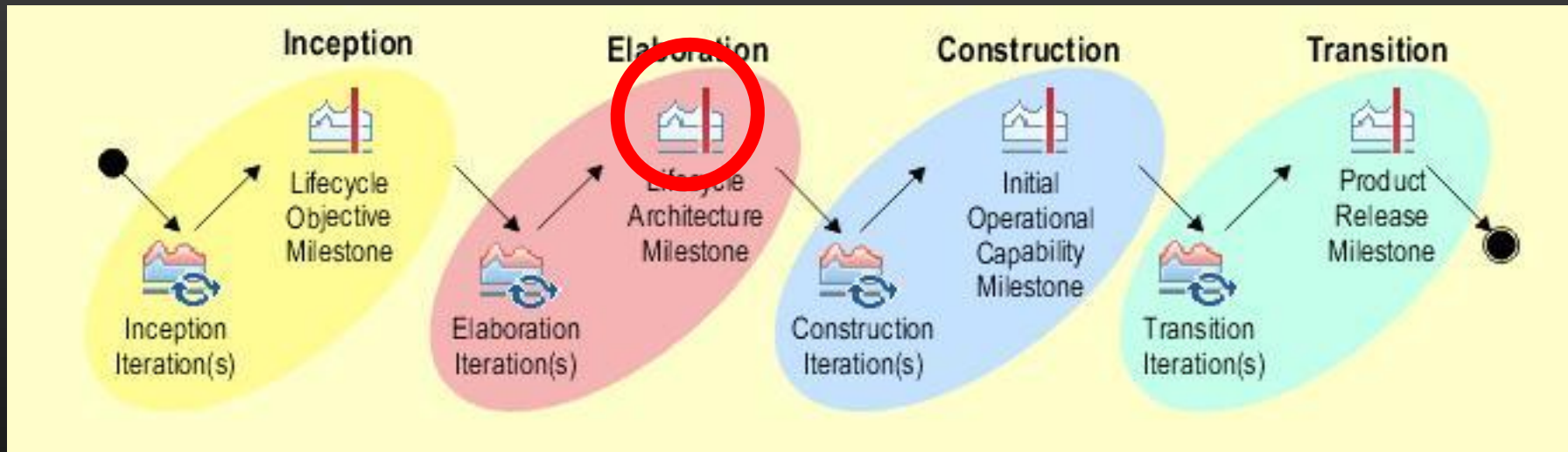
Roughly 10% of code is implemented.

Drive architecture with key use cases

20% of use cases drive 80% of the architecture

Credit: Per Kroll (IBM)

Milestones: Elaboration



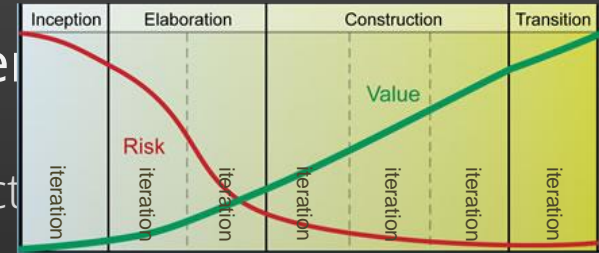
Lifecycle Architecture Milestone. At this point, a baseline of requirements is agreed to, you examine the detailed system objectives and scope, the choice of architecture, and the resolution of the major risks. The milestone is achieved when the architecture has been validated.

Construction: Build The Product

Incrementally define, design, implement more and more scenarios

Incrementally evolve executable architecture system

Evolve architecture as you go along



Frequent demonstrations and partial deployment

Partial deployment strategy depends greatly on what system you build

Daily build with automated build process

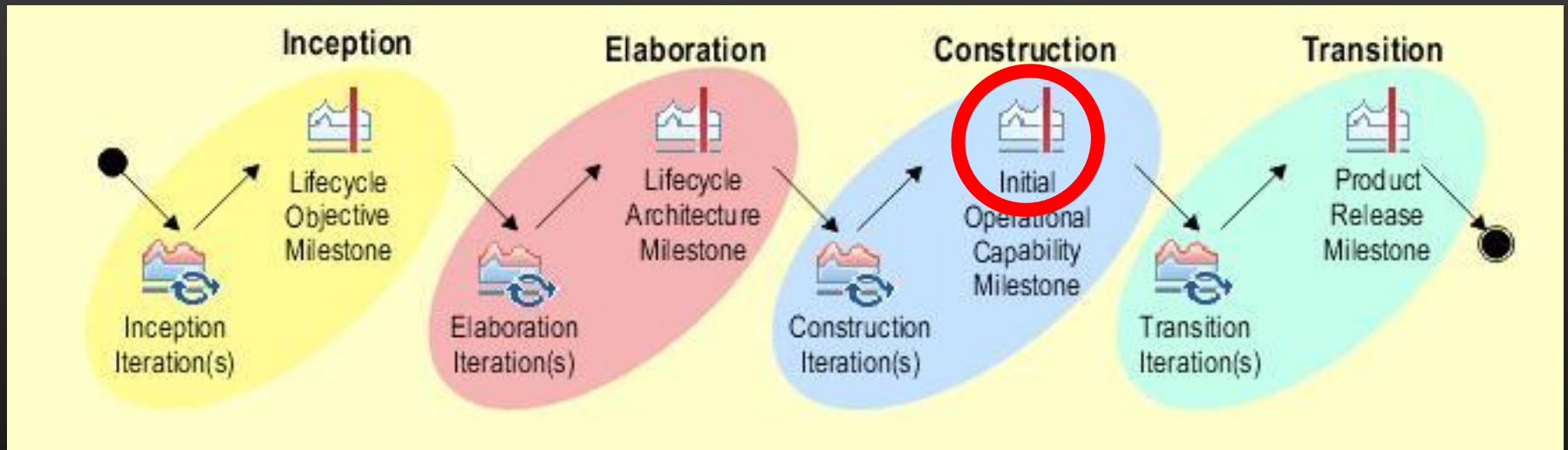
You may have to have a separate test team if you have

Complex test environments

Safety or mission critical systems

Credit: Per Kroll (IBM)

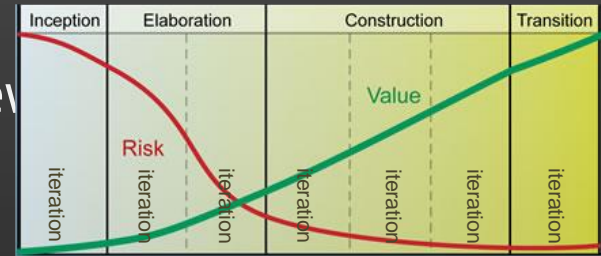
Milestones: Construction



Initial Operational Capability Milestone. At this point, the product is ready to be handed over to the transition team. All functionality has been developed and all alpha testing (if any) has been completed. In addition to the software, a user manual has been developed, and there is a description of the current release. The product is ready for beta testing.

Transition: Stabilize and Deploy

Project moves from focusing on new
stabilizing and tuning



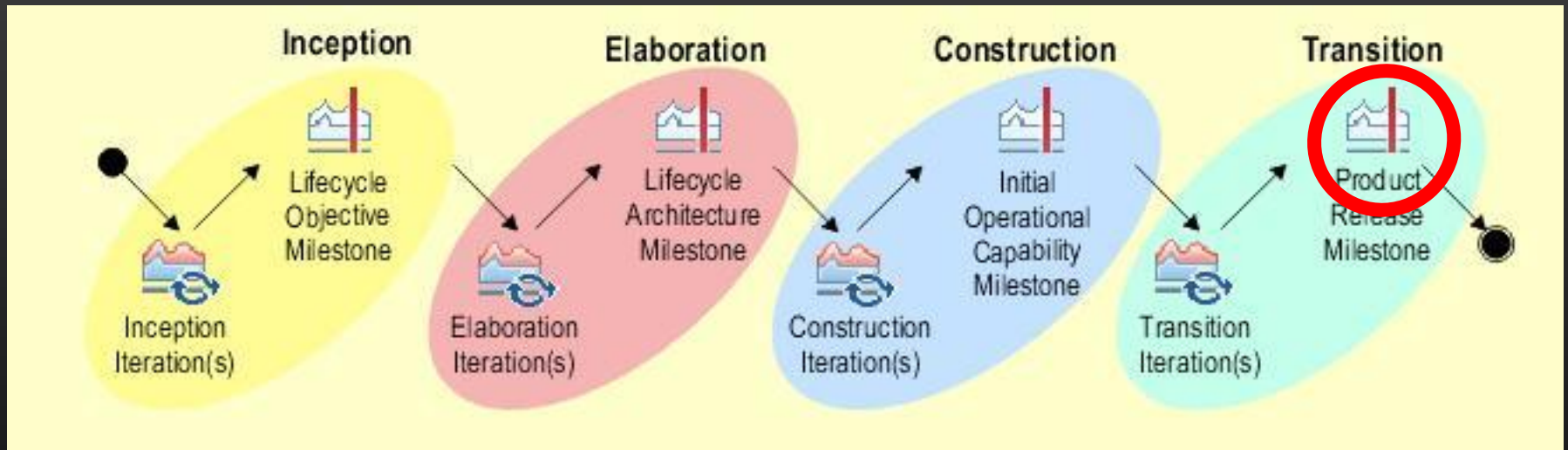
Produce incremental 'bug-fix' releases

Update user manuals and deployment documentation

Execute cut-over

Conduct "post-mortem" project analysis

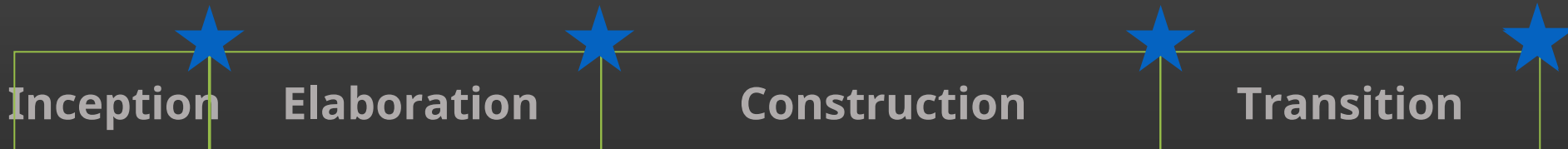
Milestones: Transition



Product Release Milestone. At this point, you decide if the objectives were met, and if you should start another development cycle. The Product Release Milestone is the result of the customer reviewing and accepting the project deliverables.

Recap main control points (lifecycle objective milestone)

Major Milestones



Time

Inception: Agreement on overall scope
Vision, high-level requirements, business case
Not detailed requirements

Elaboration: Agreement on design approach and mitigation of major risks
Baseline architecture, key capabilities partially implemented
Not detailed design

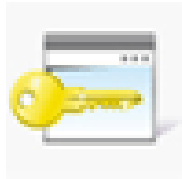
Construction: Agreement on complete operational system
Develop a beta release with full functionality

Transition: Validate and deploy solution
Stakeholder acceptance, cutover to production

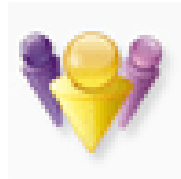
O SDLC é concretizado em **metodologias de desenvolvimento**

Adotar um processo de engenharia testado & (a)provado

O que é que inclui um processo?



Core
Principles



Roles



Work Products



Disciplines



Lifecycle

http://sweet.ua.pt/ico/OpenUp/OpenUP_v1514/

An Agile Process

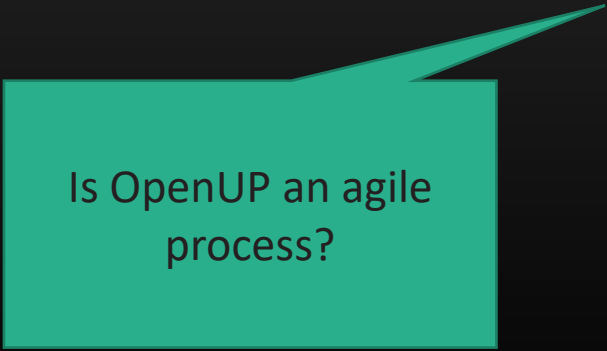
Is driven by customer descriptions of what is required (scenarios)

Recognizes that plans are short-lived

Develops software iteratively with a heavy emphasis on construction activities

Delivers multiple 'software increments'

Adapts as changes occur



Is OpenUP an agile process?

Readings & references

Core readings	Suggested readings
<ul style="list-style-type: none"><li data-bbox="150 411 884 464">• [Pressman'15] – Chap. 4, 5	<ul style="list-style-type: none"><li data-bbox="969 411 1561 464">• [Dennis'15] – Chap 1.