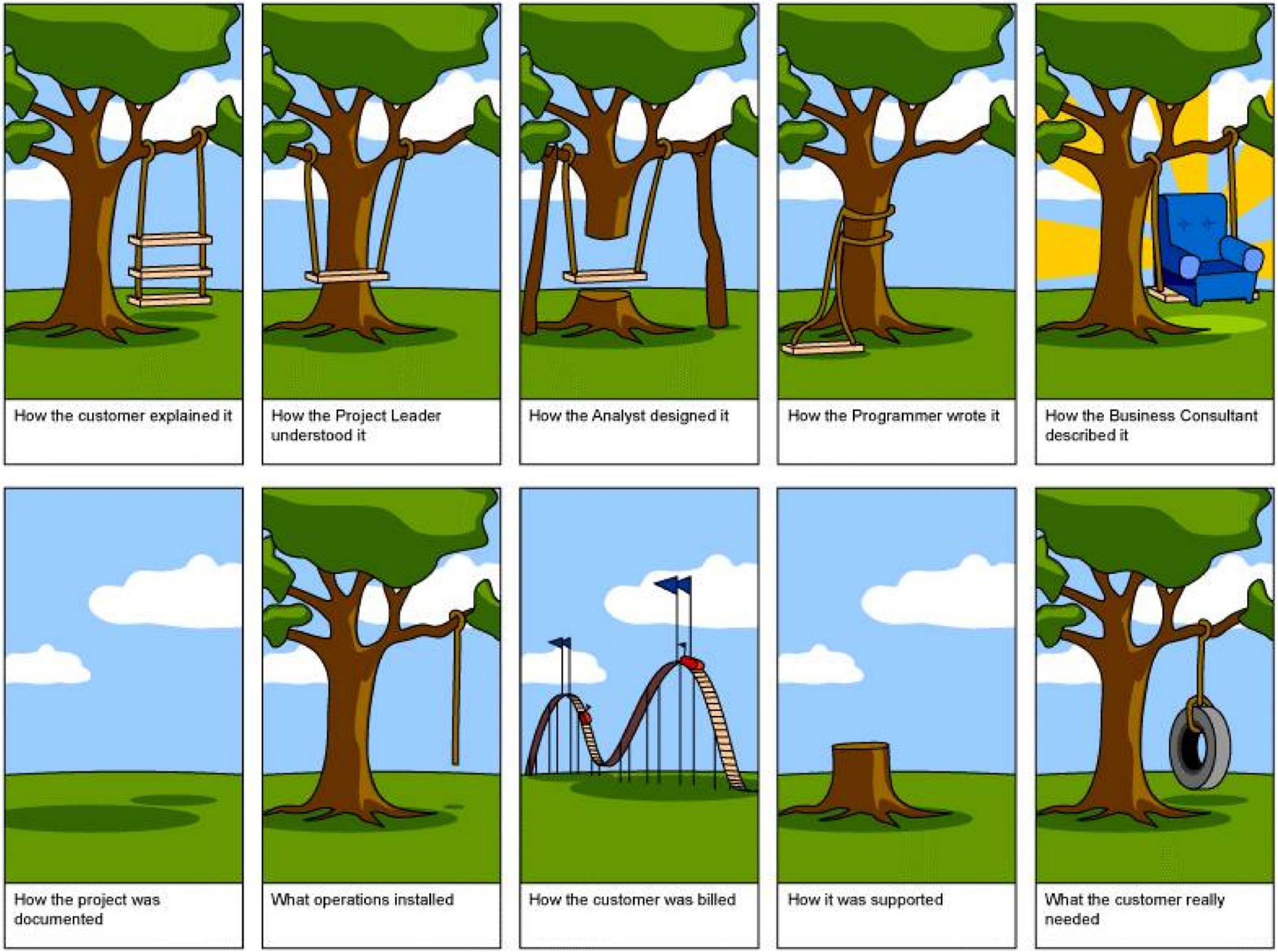
FACULDADE ENGENHARIA

Departamento de Informática

Software Process#1

(adapted from lecture notes of the CSCI 3060U - Software Quality Assurance unit, J.S. Bradbury, J.R. Cordy, 2018)


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Quality in Context

- In order to understand the roles of quality assurance in software development, we must understand how software development works – we cannot discuss inspection, and testing in a vacuum
- As background, therefore, we will begin by reviewing:
  - Major process models of the software development community
  - the ways software development efforts are organized
  - Some ways of assessing development process quality
  - Quality management standards for software processes

Software Process Models

Software Process Models

- A software development process is a method for developing computer software that organizes the effort into a number of separate tasks and steps
- This helps make it possible to develop large software systems using many people in an organized, manageable and trackable way, in order to retain control of the development

NOTE: Having control addresses QA principle 1: know what you are doing

Software Process Models

Fundamental Process Activities

- All software process models share four fundamental process activities, and differ primarily in how these four are organized and interleaved

    - Specification

define requirements, functionality and constraints

    - Development

build software to meet the specification

    - Validation

validate that it does what the customer wants

    - Evolution

evolve to meet changing needs and expectations

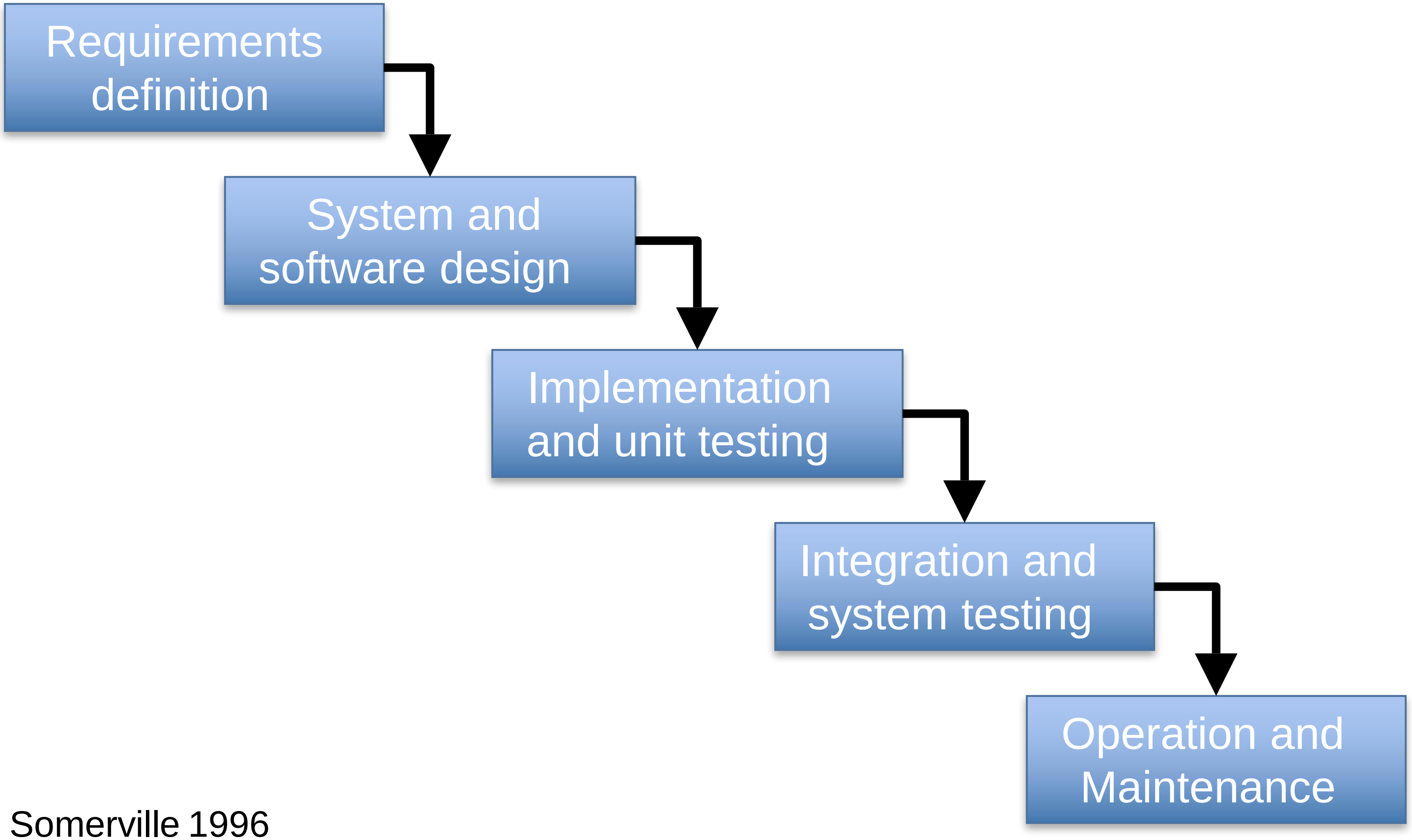
The Waterfall Model

Original Waterfall Model

- First explicit model, derived from other engineering processes
- Cascade of phases, carried out in order, with sign-off of each before proceeding to the next
- Organizes quality control, e.g., IBM's "ETVX" - Entry, Task, Validation, eXit at each step

The Waterfall Model

Original Waterfall Model



Requirements definition

System and software design

Implementation and unit testing

Integration and system testing

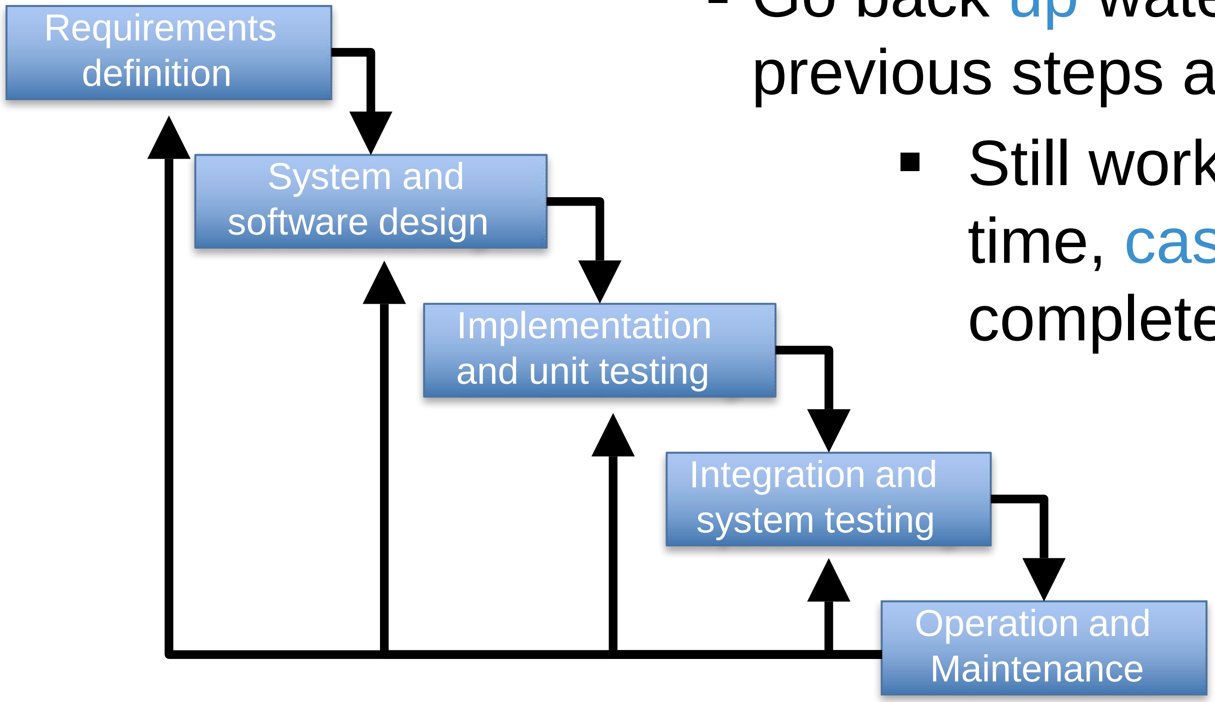
Operation and Maintenance

<u>Source</u>: Somerville 1996

The Waterfall Model

Iterative Waterfall Model

- Refined to more realistic with practice


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and Maintenance

- Go back up waterfall to revisit previous steps as necessary
  - Still work on one step at a time, cascade to next as completed

Source: Somerville 1996

Departamento de
Informática

The Waterfall Model

(1) Requirements Analysis and Definition

- System’s required services, constraints and goals are established by consultation with users/customers
- Expressed in a way understood and agreed to by both users and developers – often test cases or scenarios
  - Quality control – requirements reviews (inspection)

(2) System and Software Design

- Partitions into hardware and software subsystems
- Establishes overall system and software architecture
- Establishes functional specifications for components of the architecture
  - Quality control - design reviews (inspection)

The Waterfall Model

(3) Implementation and Unit Testing

- Design realized as a set of programs and program components (units) to implement components of the architecture
- Verify that units meet functional specifications
- Quality control - unit testing, component testing

(4) Integration and System Testing

- Integrate individual programs and program units into complete system
- Validate system that system meets requirements
- Quality control - integration testing, acceptance testing

The Waterfall Model

(5) Operation and Maintenance

- Normally longest phase of software life cycle
- Install system and put into use
- Maintenance involves correcting errors discovered in practice (*"failures"*) and improving system units (e.g., performance tuning) and enhancing services in response to new requirements
- Quality control - regression testing, acceptance testing

(6) Retirement and Decommissioning

- System is retired and replaced with a new one
- Rarely done now because of cost and risk of replacement- continuous evolution more common

Drawbacks of the Waterfall Model

Early Freezing

- In practice frequent iterations back up the waterfall make it difficult to identify checkpoints and track progress
- Therefore it is normal to freeze parts of the development, such as requirements and design, and move on to the later stages quite early without feedback
- Premature freezing of requirements may mean that the system won’t end up doing exactly what the users want
- Premature freezing of designs often leads to badly structured systems as design problems are worked around using implementation tricks

Drawbacks of the Waterfall Model

Inflexible Partitioning

- The inflexible partitioning into distinct stages, while a management advantage, often leads to undesirable technical results
- Delivered systems are sometimes unusable, do not meet users' *real* requirements (as opposed to their original guesses)

But …

- The waterfall model reflects common engineering practice
- Likely that this process model will still remain the norm for some time

The Prototyping Model

Problems with Requirements

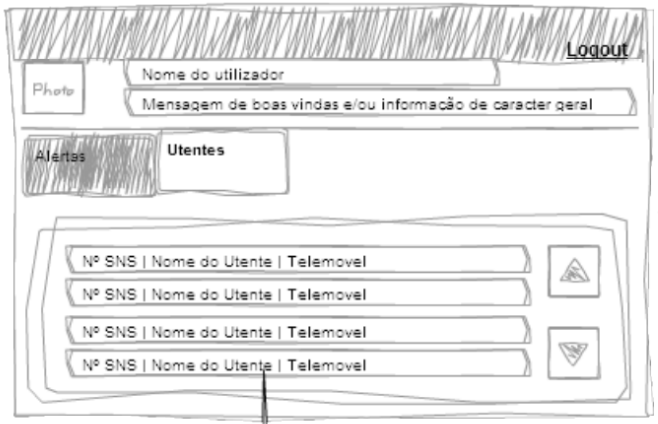
- The first step in the waterfall is requirements gathering and analysis
- In practice, this is the most difficult part and experience with the waterfall – indicates that most failures are due to inadequate requirements understanding
- Users often change requirements as they see what can be done

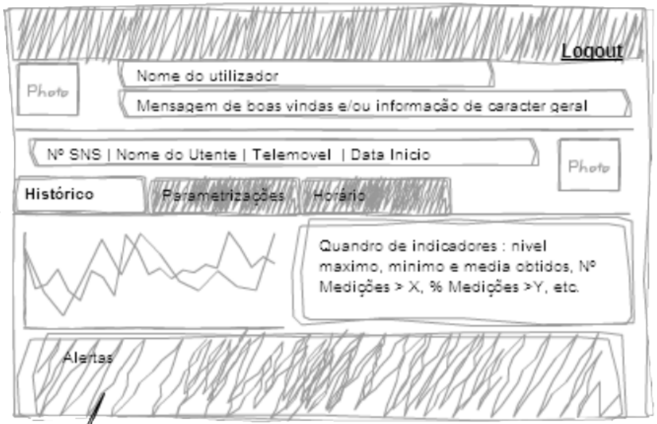
The Prototyping Model

Prototyping

- The prototyping model attempts to address the requirements difficulty by introducing an iterative, by example requirements stage
- A prototype is a partial implementation of a software system with all external interfaces presented
- Users use the prototype and provide feedback from which real requirements are gradually refined
- Final prototype serves as example of intended system

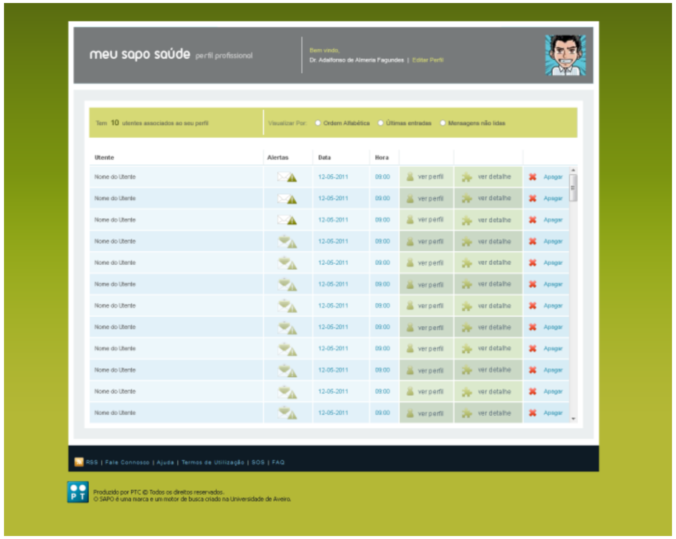
Prototyping examples


Logout
Photo
Nome do utilizador
Mensagem de boas vindas e/ou informação de caracter geral
Alertas
Utentes
Nº SNS | Nome do Utente | Telemovel
Nº SNS | Nome do Utente | Telemovel
Nº SNS | Nome do Utente | Telemovel
Nº SNS | Nome do Utente | Telemovel


Logout
Photo
Nome do utilizador
Mensagem de boas vindas e/ou informação de caracter geral
Nº SNS | Nome do Utente | Telemovel | Data Inicio
Photo
Histórico
Parametrizações
Horário
Quandro de indicadores : nivel maximo, minimo e media obtidos, Nº Medições > X, % Medições >Y, etc.
Alertas

Listagem de utentes associados ao Profissional

Listagem com histórico de alertas do Utente.

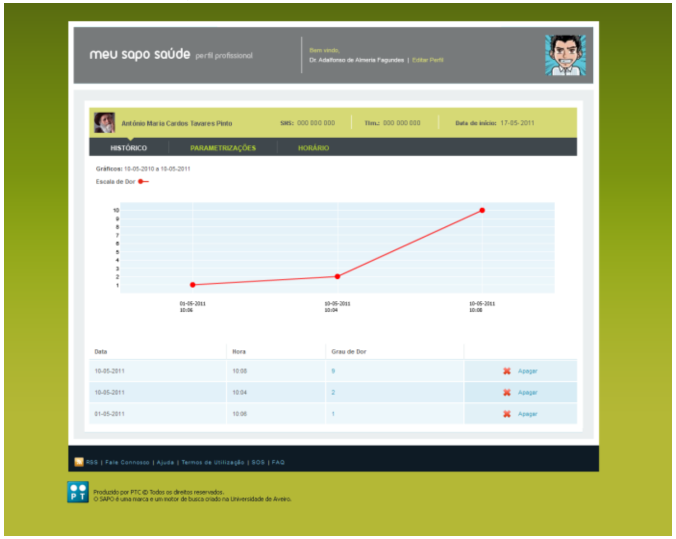

meu sapo saúde perfil profissional
Bem vindo,
Dr. Adalfonso de Almeria Fagundes | Editar Perfil

Tem 10 utentes associados ao seu perfil | Visualizar Por: ○ Ordem Alfabética ○ Últimas entradas ○ Mensagens não lidas

Utente	Alertas	Data	Hora			
Nome do Utente		12-05-2011	09:00	ver perfil	ver detalhe	✖ Apagar
Nome do Utente		12-05-2011	09:00	ver perfil	ver detalhe	✖ Apagar
Nome do Utente		12-05-2011	09:00	ver perfil	ver detalhe	✖ Apagar
Nome do Utente		12-05-2011	09:00	ver perfil	ver detalhe	✖ Apagar
Nome do Utente		12-05-2011	09:00	ver perfil	ver detalhe	✖ Apagar
Nome do Utente		12-05-2011	09:00	ver perfil	ver detalhe	✖ Apagar
Nome do Utente		12-05-2011	09:00	ver perfil	ver detalhe	✖ Apagar
Nome do Utente		12-05-2011	09:00	ver perfil	ver detalhe	✖ Apagar
Nome do Utente		12-05-2011	09:00	ver perfil	ver detalhe	✖ Apagar
Nome do Utente		12-05-2011	09:00	ver perfil	ver detalhe	✖ Apagar
Nome do Utente		12-05-2011	09:00	ver perfil	ver detalhe	✖ Apagar
Nome do Utente		12-05-2011	09:00	ver perfil	ver detalhe	✖ Apagar

RSS | Fale Connosco | Ajuda | Termos de Utilização | SOS | FAQ

Produzido por PTC © Todos os direitos reservados.
O SAPO é uma marca e um motor de busca criado na Universidade de Aveiro.



António Maria Cardos Tavares Pinto | SNS: 000 000 000 | Tlm.: 000 000 000 | Data de início: 17-05-2011

Gráficos: 10-05-2010 a 10-05-2011

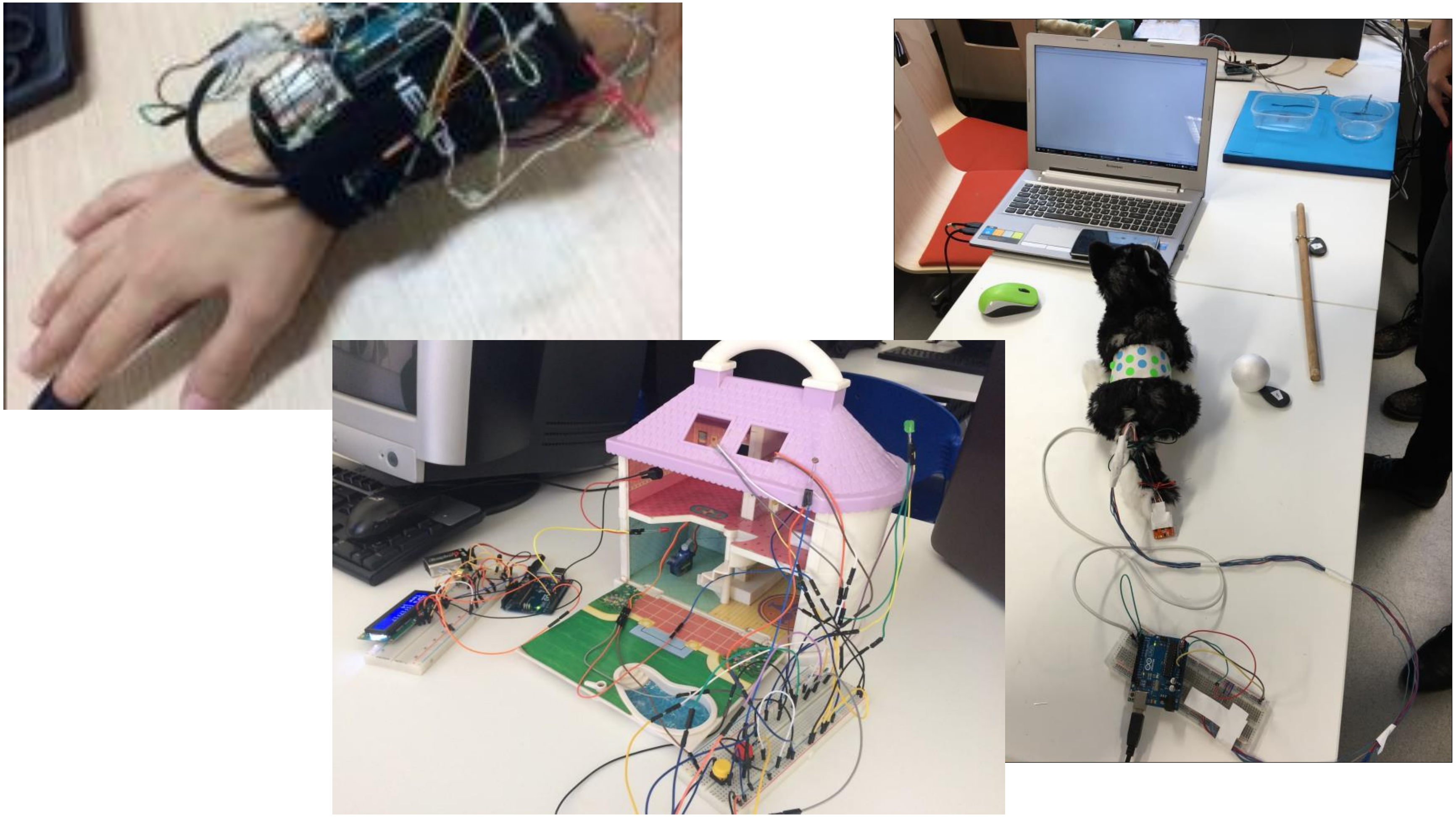
Escala de Dor ●—

Data	Hora	Grau de Dor	
10-05-2011	10:08	9	✖ Apagar
10-05-2011	10:04	2	✖ Apagar
01-05-2011	10:06	1	✖ Apagar

RSS | Fale Connosco | Ajuda | Termos de Utilização | SOS | FAQ

Produzido por PTC © Todos os direitos reservados.
O SAPO é uma marca e um motor de busca criado na Universidade de Aveiro.

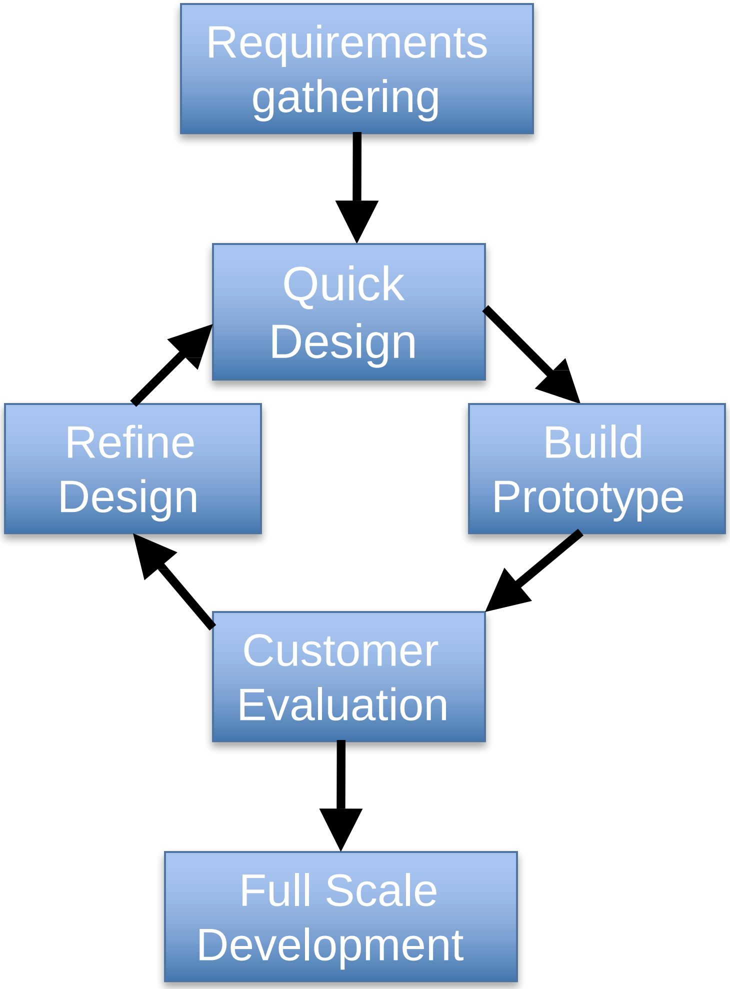
Prototyping examples



The Prototype Model

Prototyping Model

- Extend requirements phase to include a sequence of prototypes
- Improve requirements and design as prototypes refined
- When users and developers both satisfied, move on to real development



FACULDADE ENGENHARIA

Departamento de Informática

The Prototype Model

(1) Requirements Gathering and Analysis

- Much like waterfall model, but less stringent since prototype will help expose inadequacies
- Quality control – requirements reviews (inspection)

(2) Quick Design

- Make a simple approximate initial design, refine during prototype iteration
- Quality control – prototype testing

(3) Build Prototype

- Quickly hack together an approximate implementation showing salient external features
- Quality control – essentially none

The Prototype Model

(4) Customer Evaluation

- Users validate prototype, report inadequacies
- Quality control – acceptance testing and evaluation (inspection)

(5) Design Refinement

- Refine design in response to user feedback from prototype
- Quality control – design reviews (inspection)

(6) Full Scale Development

- Remaining stages of traditional waterfall model

FACULDADE ENGENHARIA

Departamento de Informática

Drawbacks of the Prototyping Model

Wasted Work

- Prototypes are normally built using substandard quality controls (*“thrown together”*) in order to speed the iteration (*“quick turnaround”*)

- Thus they must be discarded after the prototyping phase, even if they solve significant problems

Drawbacks of the Prototyping Model

Inadequate or Incomplete Prototypes

- Full prototypes of complex systems can be difficult or impossible to create quickly
- Thus prototypes are often done in parts, which may miss critical requirements at the integration or complete system stage

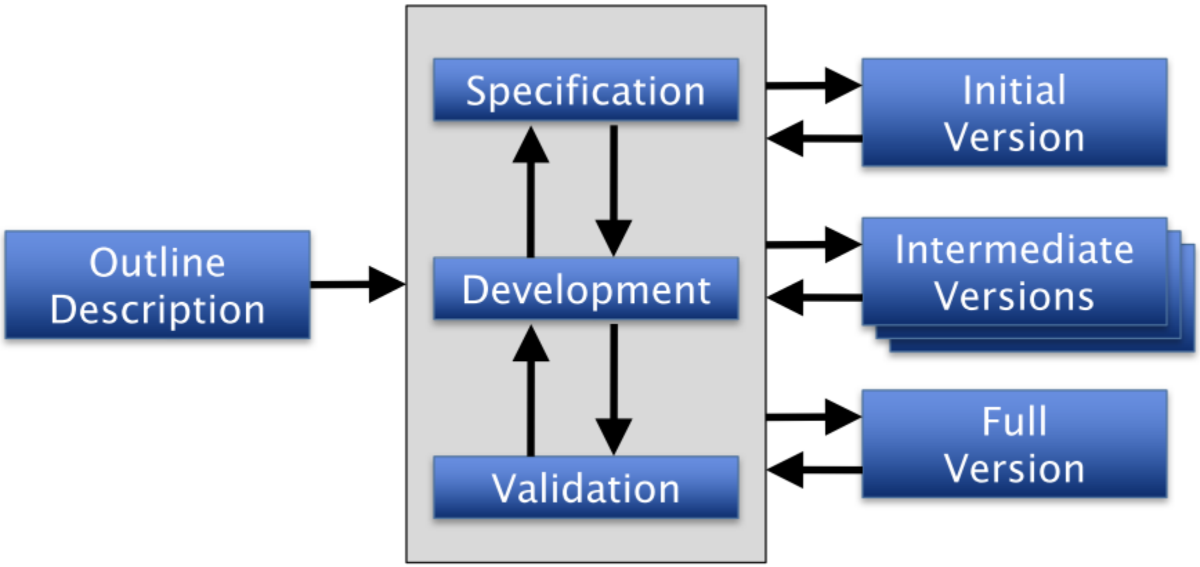
When to Stop Iterating

- Easy to have users convince you to continue refining beyond the point where requirements and design are sufficient (*“creeping excellence”*)

Evolutionary Development

Prototype Evolution

- Evolutionary prototyping is a method to avoid wasting work and take advantage of *"creeping excellence"* by smoothly evolving the initial prototype to the final product

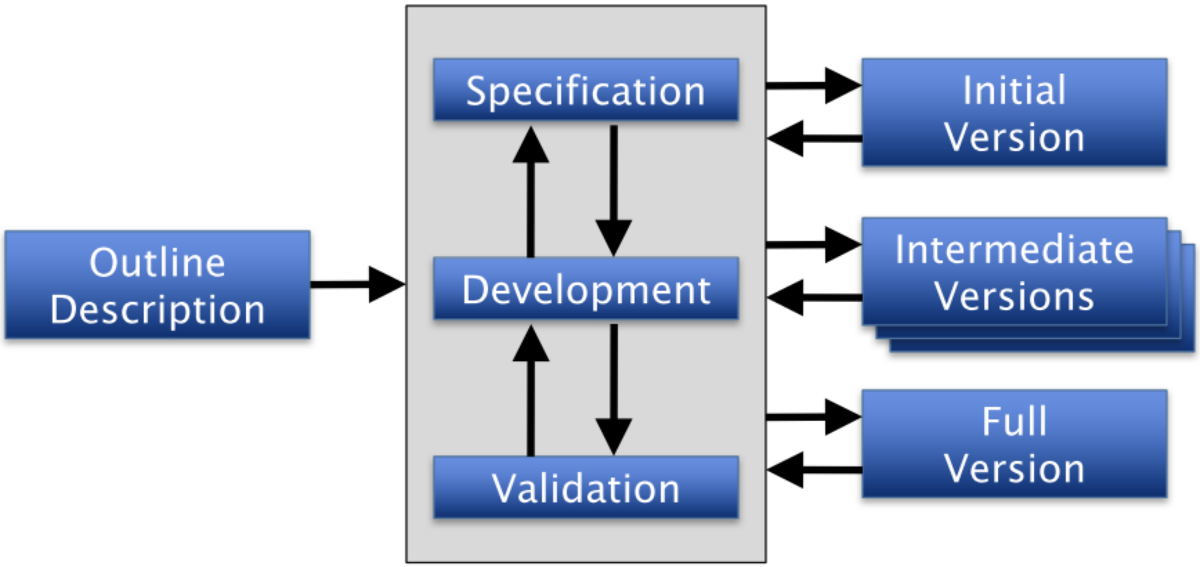


Concurrent activities

Evolutionary Development

Prototype Evolution

- In essence, never leave prototype iteration until implementation is complete



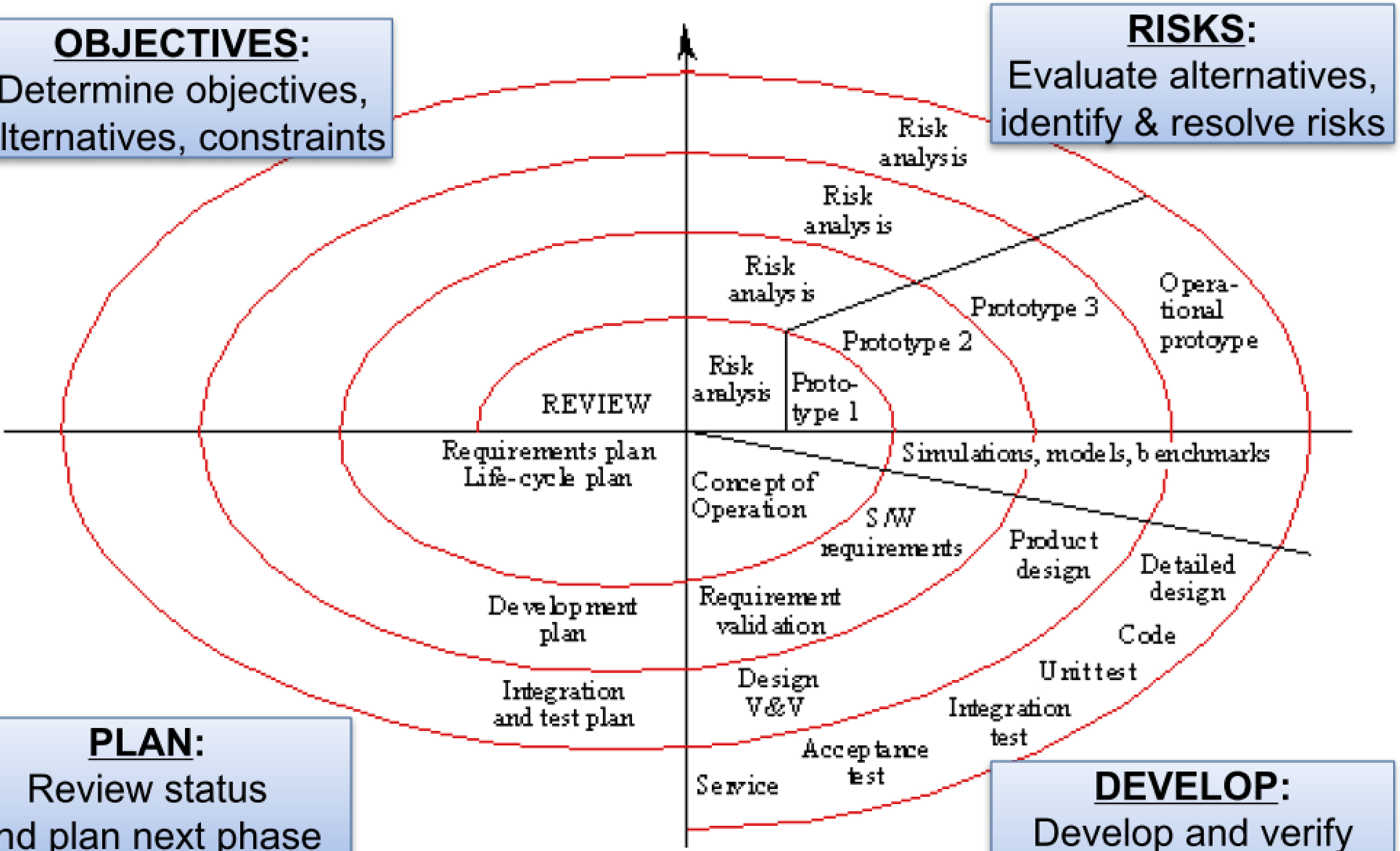
Concurrent activities

The Spiral Model

Boehm’s Spiral Model

- The Spiral Model is a refinement of the waterfall model designed around continuous documentation and evaluation of risk
- Based on experience applying the waterfall model to large software projects (e.g. government projects)
- Now a standard used by many government agencies and software providers

The Spiral Model



OBJECTIVES:
Determine objectives, alternatives, constraints

RISKS:
Evaluate alternatives, identify & resolve risks

PLAN:
Review status and plan next phase

DEVELOP:
Develop and verify next level product

Source: Boehm 1988

The Spiral Model

Spiral Layers

- Roughly, each layer of the spiral corresponds to one phase of the waterfall (although there are no fixed phases)

- For example, first layer could be the requirements phase, second layer the design phase, etc.

The Spiral Model

Four Step Cycle

- In each layer, the same four step cycle is used, consisting of:

  - Determine Objectives

determine objectives, constraints, risks for next phase

  - Assess and Reduce Risks

analyze and reduce identified risks

  - Develop and Validate

choose development model, develop and test

  - Review and Plan

review status, plan next layer

The Spiral Model

- For each layer (phase) of the project:

(1) Determine Objectives

- Specific objectives (aims) for the phase of the project are defined
- Constraints on the process and product are identified
- Alternatives for achieving the objectives are identified
- Potential risks associated with each alternative are identified

(2) Assess and Reduce Risks

- For each potential risk, a detailed analysis is carried out
- Steps are taken to reduce risk (e.g., create prototype to check)
- Alternatives are chosen to minimize risk

The Spiral Model

(3) Develop and Validate

- Based on risk analysis, choose or modify development model
- For example, to implement and validate,
  - if user interface risks dominate, use evolutionary prototyping
  - if safety risks are the major issue, use formal methods
  - if integration problems are the big risk, use waterfall model

(4) Review and Plan

- Review and evaluate results of this phase (layer)
- Decide whether another layer of the spiral is needed
- Draw up plans for next phase if so

The Spiral Model

Heavyweight Process

- The spiral model requires a large amount of overhead – every layer requires a lot of documentation and many meetings – progress can therefore be slow
- Primarily suitable for large projects with long timelines

Not Really a Development Model

- The spiral model is really more of a *“meta-model”* since it describes the way to carry out stages, not what the stages are
- But focuses on identifying potential problems early at every stage, so very good at producing high quality results

The Spiral Model

Depends on Risk Analysis

- Needs a very experienced team to recognize and analyse risks accurately
- High dependency on quality of people (itself a risk!)

Not for Novices

- Layers of process are flexible and not explicitly laid out
- Each layer's goals and plan must be decided by team itself – requires experienced people

The Iterative Development Model

Subset Development

- The Iterative Development Process (IDP) is based on subsets
- Begin with a subset of the requirements and develop a subset of the software product
- The subset should:
  - satisfy immediate needs of users
  - serve as a vehicle of training for customers, and learning for developers

The Iterative Development Model

Sequence of Intermediate Products

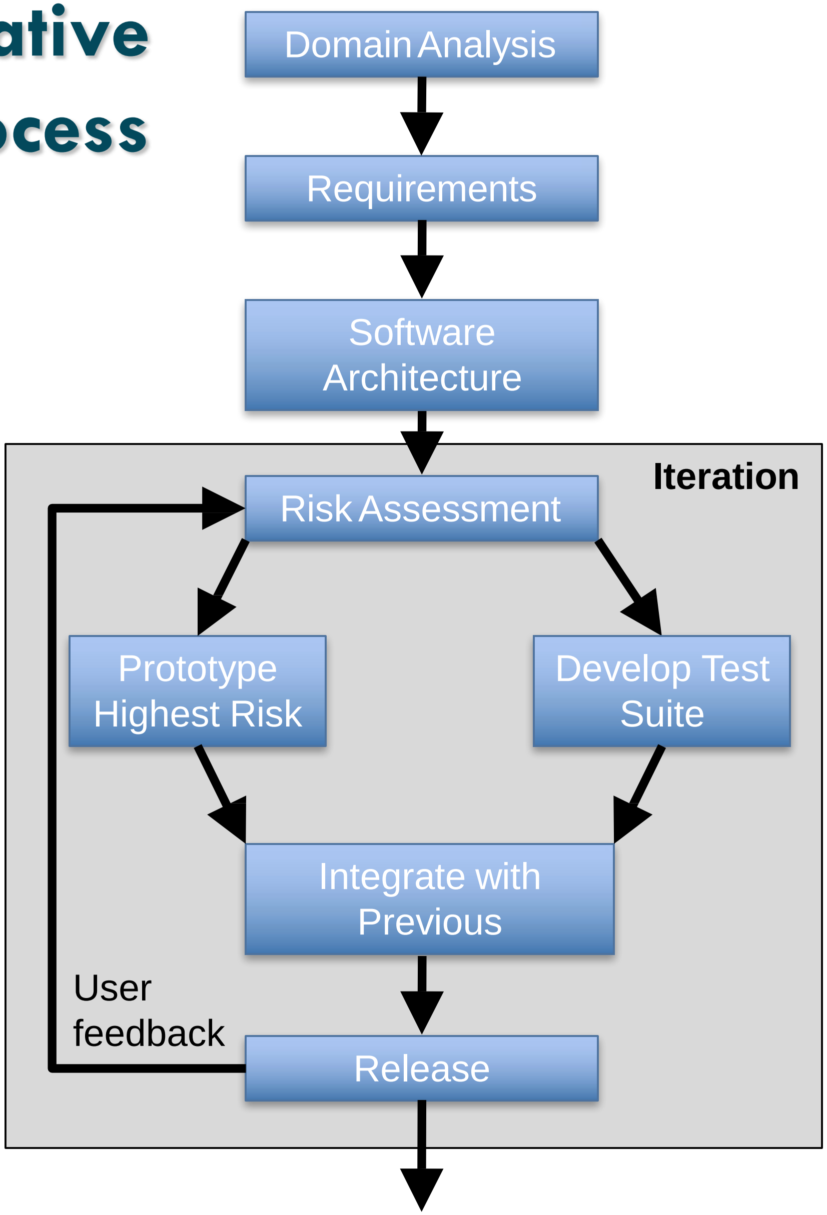
- Analysis of the subset product leads to modifications to the design and requirements, from which we build a new (hopefully larger) subset product

- Design and requirements refined over a series of iterations to provide a system that meets evolving customer needs with improved design based on feedback and learning

The Iterative Development Process

Iterative Development Process

- Analysis of the problem domain and definition of requirements begins process as usual
- Need initial architecture design to begin
- Add most critical remaining features each cycle
- Quality Control: development of test suite for new features on each interaction



Drawbacks of the Iterative Process

Needs Small Team

- Process does not allow for large scale parallel development, depends on focussing on one remaining risk at a time
- Works best with relatively small teams

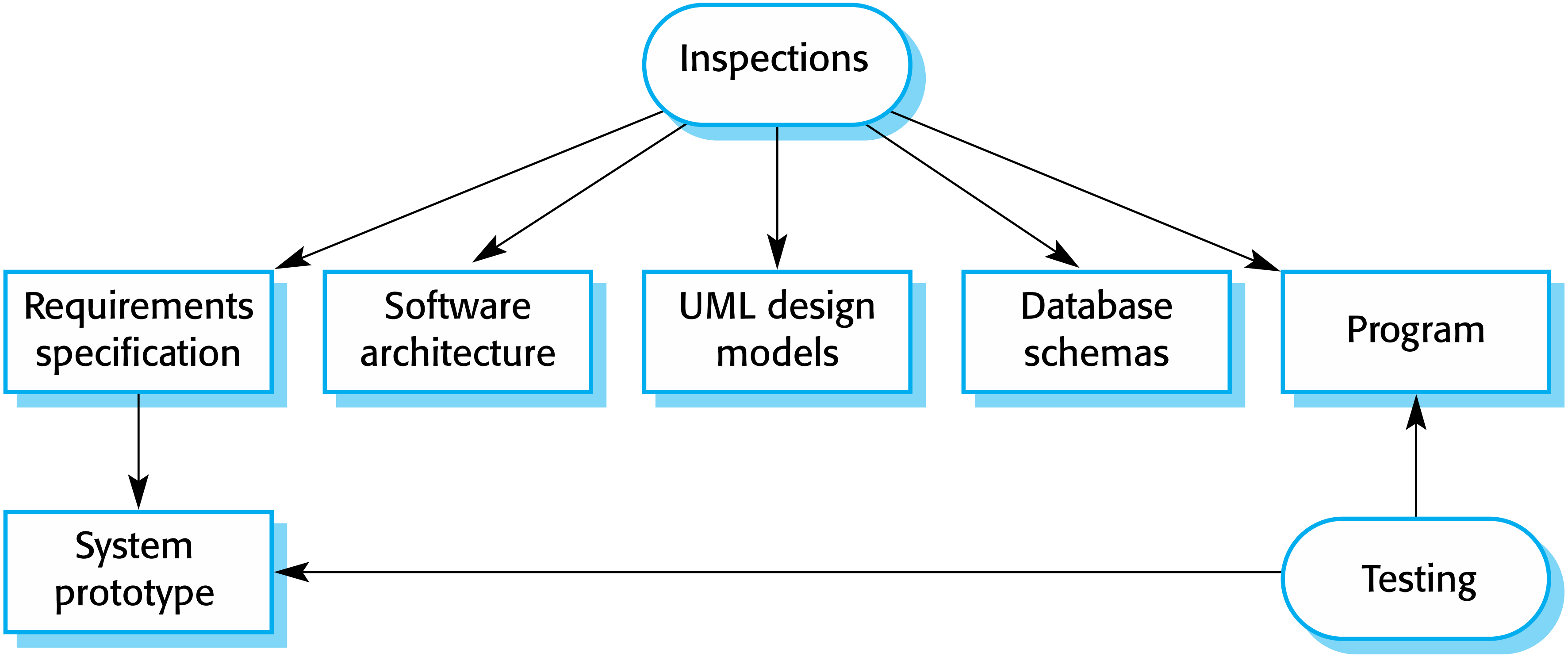
Needs Early Architecture

- Requires early design of overall architecture, difficult to change later
- But when architecture can be settled early, has been very successful at producing significant, very high quality products, e.g., IBM’s OS/2 system

Inspections and Testing

- **Software inspections**, concerned with analysis of the static system representation to discover problems (static verification)
  - May be supplement by tool-based document and code analysis.
- **Software testing**, concerned with exercising and observing product behaviour (dynamic verification)
  - The system is executed with test data and its operational behaviour is observed.

Inspections and Testing



Software Inspections

- These involve people examining the source representation with the aim of discovering anomalies and defects.
- Inspections not require execution of a system so may be used before implementation.
- They may be applied to any representation of the system (requirements, design, configuration data, test data, etc.).
- They have been shown to be an effective technique for discovering program errors.

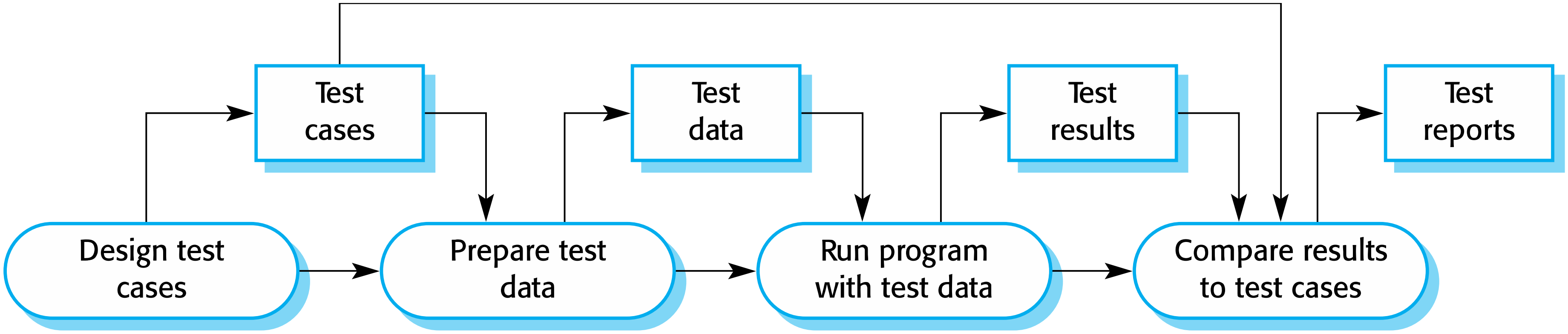
Advantages of Inspections

- During testing, errors can mask (hide) other errors. Because inspection is a static process, you don’t have to be concerned with interactions between errors.
- Incomplete versions of a system can be inspected without additional costs. If a program is incomplete, then you need to develop specialized test harnesses to test the parts that are available.
- As well as searching for program defects, an inspection can also consider broader quality attributes of a program, such as compliance with standards, portability and maintainability.

Inspections and Testing

- Inspections and testing are complementary and not opposing verification techniques.
- Both should be used during the Verification & Validation process.
- Inspections can check conformance with a specification but not conformance with the customer’s real requirements.
- Inspections cannot check non-functional characteristics such as performance, usability, etc.

A model of the software testing process



Stages of Testing

- Development testing, where the system is tested during development to discover bugs and defects.
- Release testing, where a separate testing team test a complete version of the system before it is released to users.
- User testing, where users or potential users of a system test the system in their own environment.

Summary

- Software development has four tasks
- Software development processes differ in how these are interlaced
- Oldest and most common process is the Waterfall Process
- Some recent and popular processes are based on Prototyping
- Spiral Model organizes and generalizes waterfall model
- Iterative Development Process based on product subsets
- A software inspection is the analysis of the static system representation to discover problems
- A software testing is the exercising and observing product behaviour

HOW
WHY
HOW
WHEN
WHERE
WHO
WHERE
WHERE
WHY
WHO
WHEN
WHAT
WHAT
WHO
WHY
WHO
WHAT
WHAT
WHEN
HOW
WHAT
WHO
WHEN
WHERE
WHY
HOW