ANÁLISE E CONCEÇÃO E SISTEMAS DE INFORMAÇÃO

2024/25

Francisco Duarte

francisco.duarte@dsi.uminho.pt

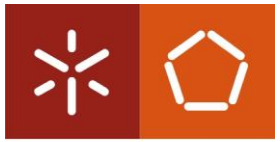
Universidade do Minho
Escola de Engenharia

DADOS GERAIS: PLANEAMENTO

1	19-set-25	Introdução ACSI		matérias a leccionar referências esquema das aulas avaliações		
2	25-set-25	Os Sistemas de Informação e a Engenharia de Software	Sistemas de informação e engenharia de software -Sistemas sócio-técnicos -Projecto, sistemas e desenvolvimento -Analise, requisitos e especificação -Referenciais SWEBoK v4	Perspectivas SWEBOK v4 processos: Scrum ou OpenUP introdução do Tema/Problema para os trabalhos		
3	2-out-25	Requisitos de Software	Engenharia e gestão de requisitos -Processo de engenharia de requisitos -Levantamento de requisitos (elicitation) -Prioritisação de requisitos -Negociação de requisitos -Escrita de requisitos	Ferramentas de trabalho Requisitos de Software Processos de Engenharia de Requisitos	M1: Technical Specification	Visual Paradigm
4	9-out-25			Levantamento de requisitos	M1: Technical Specification	Visual Paradigm
5	16-out-25			Negociação de requisitos Escrita de requisitos	M1: Technical Specification Interacção dos grupos com o cliente TUB (via Zoom)	Visual Paradigm
6	23-out-25	M1 - requisitos	apresentação Trabalho 1		Technical Specification (OpenUP: Glossary; Vision; System-wide Requirements; Use-case Model; Use	
7	30-out-25	Arquitetura de Software	Arquitectura de Software -Padrões -Estilos arquiteturais -Classificação de padrões -Catálogos de padrões -Representação de arquitecturas -Arquitecturas empresariais (acc. SWEBOK v4) Arquitetura lógica (4SRS) -Referenciais EABoK, TOGAF	Conceitos e definições (acc. SWEBOK v4) Arquitetura lógica (4SRS)	M2: Architecture 4SRS	4SRS Excel
8	6-nov-25			Architecture Views e Viewpoints Reactive Manifesto modelação da arquitectura (UML)	M2: Architecture UML: classes, sequencias, e outros diagramas de arquitectura	Visual Paradigm
9	13-nov-25			Architectural Styles and Patterns Software Architecture Evaluation EABOK / TOGAF	M2: Architecture	Visual Paradigm
10	20-nov-25	M2: arquitetura + revisão requisitos	apresentação Trabalho 2	CPES/SERIW 20/11	Architecture Notebook; Arquitetura lógica 4SRS, Arch Patterns	
11	21-nov-25	Conceção de Software	Conceção de Software -Design patterns -MDA -frameworks de implementação: Neadvance Niop + Apache Camel e Apache Kafka	Conceitos e definições (acc. SWEBOK v4)	M3: conceção detalhada dos módulos a implementar, respeitando a arquitectura proposta	Visual Paradigm
12	4-dez-25			Design Patterns Pittfalls (anti-patterns) Visita Neadvance: explicação e exemplos low-code usando niop?	M3: implementação recorrendo a Neadvance Niop e Apache Camel + Apache Kafka	Neadvance Niop + Java IDE + Apache Camel + Apache Kafka
13	11-dez-25			MDA	M3: implementação recorrendo a Neadvance Niop e Apache Camel + Apache Kafka	Neadvance Niop + Java IDE + Apache Camel
14	18-dez-25	M3: design + rev arquitetura e requisitos	apresentação Trabalho 3		OpenUP: design; Design Patterns; Neadvance niop? + Apache Camel + Apache Kafka	

Software Architecture
Definições e Conceitos

Francisco J. Duarte



Universidade do Minho

Departamento de Sistemas de Informação

Definitions

The software architecture of a system is the set of structures needed to reason about the system. These structures comprise software elements, relations among them, and properties of both.

L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 4th edition, 2021.

Contexto

1. A **Arquitectura de Software** está **focada** no que é **fundamental** para o sistema de software: nem todos os elementos, interconexões, ou interfaces são fundamentais;
2. A **Arquitectura de Software** **considera** o sistema no seu **ambiente**: o contexto fora das fronteiras do sistema é considerado através das pessoas, organizações, software, hardware, e outros dispositivos com os quais interage.

ISO/IEC/IEEE 42010:2011, Systems and software engineering – Architecture description.

Utilidade

Uso principal: dar uma **compreensão compartilhada** acerca do sistema para **guiar** a sua **concepção** (design) e **construção**.

Também serve:

-como uma **concepção preliminar** do sistema de software de modo a servir de base para **analisar** e **avaliar alternativas**;

-permitir "**arquitectura reversa**", i.e. ajuda os profissionais na manutenção, melhorias, ou modificações a um sistema já existente.

SWEBOK v4 Draft (consultado em 05/10/2022)

Preocupações (concerns)

Um sistema de software tem vários **interessados** (*stakeholders*) com variados **papéis** e **interesses** no sistema. Esses interesses são as **preocupações arquitecturais(*concerns*)**.

Exemplos de *concerns*:

affordability, agility, assurance, autonomy, **availability**, behaviour, **business goals** and strategies, **complexity**, compliance with regulation, concurrency, control, cost, data accessibility, deployability, disposability, **energy efficiency**, evolvability, extensibility, feasibility, flexibility, functionality, information assurance, inter-process communication, **interoperability**, known limitations, **maintainability**, modifiability, modularity, openness, performance, **privacy**, quality of service, reliability, resource utilization, reusability, safety, **scalability**, schedule, security, system modes, **software structure**, subsystem integration, sustainability, system features, testability, **usability**, usage, user experience.

SWEBOK v4 Draft (consultado em 05/10/2022)

Separation of Concerns (Dijkstra):

focar num *concern* de cada vez (não quer dizer que os outros *concerns* sejam ignorados).

Software Architecture

(tópicos SWEBOK v4.0a)

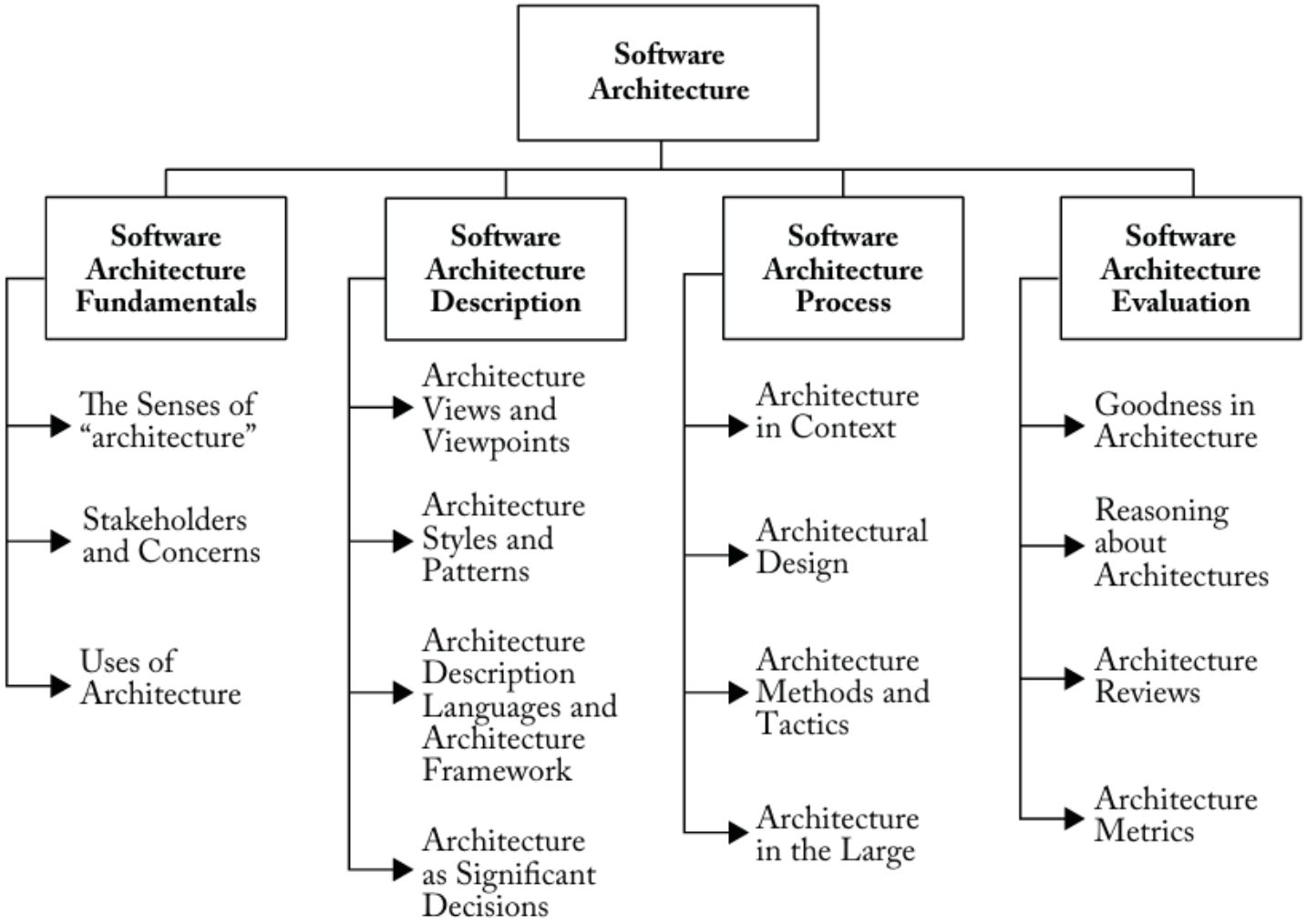
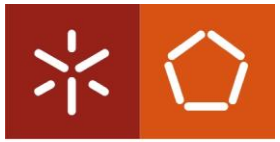


Figure 2.1. Breakdown of Topics for the Software Architecture KA

Software Architecture
4SRS

Francisco J. Duarte



Universidade do Minho

Departamento de Sistemas de Informação

4SRS

Propósito:

Técnica para transformar modelos de requisitos funcionais em modelos de arquitectura (lógica).

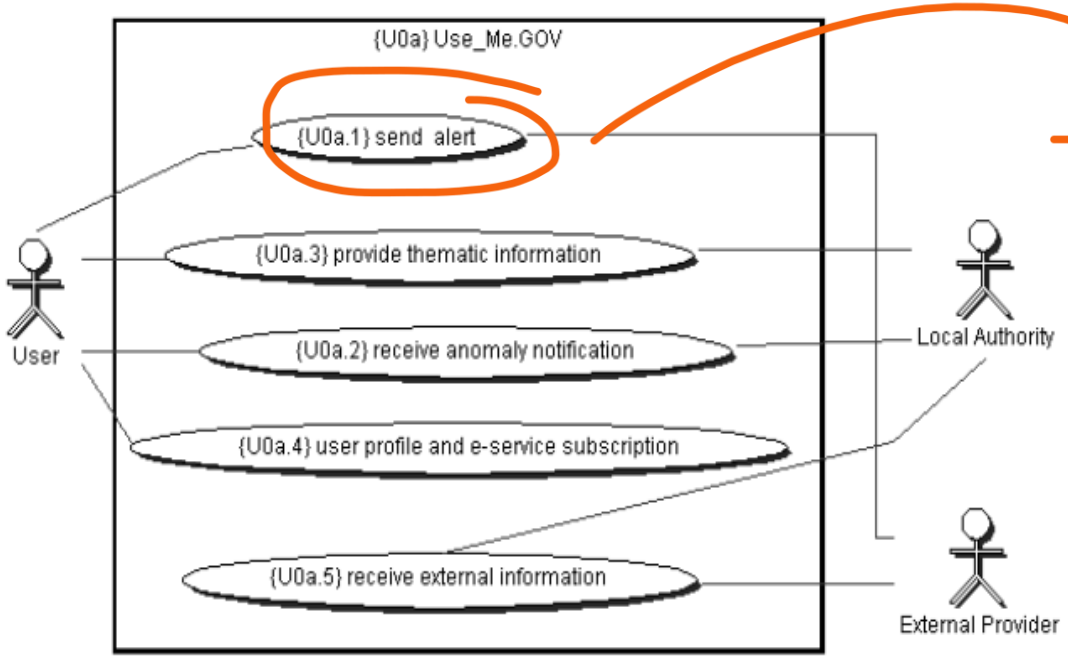
R. Machado, J. Fernandes, P. Monteiro, H. Rodrigues, Transformation of UML Models for Service-Oriented Software Architectures, ECBS'05, 2005.

4SRS

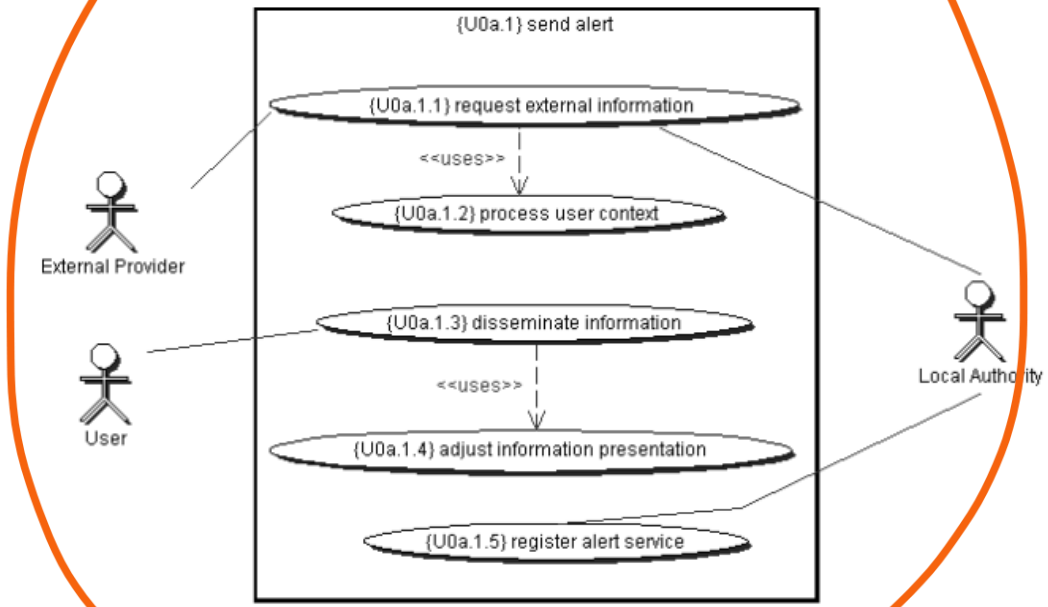
Modelo de Requisitos

UML/diagrama de casos de uso refinados

Modelo abstrato de casos de uso:



Modelo do caso de uso U0a.1 refinado:



4SRS

Passo 1: Criação de objectos

Para cada caso de uso refinado criam-se três objectos, com os sufixos:

i-interface

c-controlo

d-dados

Exemplo:

uc1.1.1 -> uc1.1.1.i, uc1.1.1.c, uc1.1.1.d

4SRS

Passo 2: Eliminação de objectos

Micropasso 2i: classificação dos casos de uso

Cada **caso de uso refinado** é classificado como uma das opções: ∅, i, c, d, ic, di, cd, icd

Exemplo:

uc1.1.1 -> di

4SRS

Passo 2: Eliminação de objectos

Micropasso 2ii: eliminação local

Baseado na classificação feita no micropasso 2i, os objectos desnecessários são eliminados

Exemplo:

uc1.1.1 (classificado como di) -> o objecto uc1.1.1.c é eliminado

4SRS

Passo 2: Eliminação de objectos

Micropasso 2iii: nomeação de objectos

Os objectos que não foram eliminados no passo anterior recebem um nome que reflete o caso de uso que o originou e o papel que desempenha no sistema

Exemplo:

uc1.1.1.i -> "serviço de registo de cliente"

Uc1.1.1.d -> "cadastro de cliente"

4SRS

Passo 2: Eliminação de objectos

Micropasso 2iv: descrição de objectos

Cada objecto nomeado deve ser descrito de forma a que os requisitos que representa (vindos dos casos de uso, ou requisitos não funcionais, ou decisões de concepção) sejam incluídos no modelo de objectos.

Exemplo:

"cadastro de cliente": este objecto guarda os atributos do interface com o utilizador de registo de clientes. Deve guardar atributos que permitam a descoberta, agregação, activação, e acesso a clientes.

NFR1: objecto distribuído em base regional em Portugal.

NFR2: escalável até 10M de clientes;

DC1: implementado com microserviços

4SRS

Passo 2: Eliminação de objectos

Micropasso 2v: representação de objectos

Cada objecto deve ser avaliado acerca da possibilidade de ser representado por outro objecto.

Exemplo:

"actualização de cliente" -> pode ser representado por outro objecto existente, e.g. "cadastro de cliente "

4SRS

Passo 2: Eliminação de objectos

Micropasso 2vi: eliminação global

Cada objecto representado por outro no passo anterior deve ser eliminado.

Exemplo:

"actualização de cliente" -> é eliminado, pois fica representado pelo objecto "cadastro de cliente"

4SRS

Passo 2: Eliminação de objectos

Micropasso 2vii: renomeação de objectos

Cada objecto não eliminado no passo anterior deve ser renomeado de forma a expressar todos os objectos que representa.

Exemplo:

Como o objecto "cadastro de cliente" também representa o objecto "actualização de cliente", podemos renomear "cadastro de cliente" para "ficha de cliente"

4SRS

Passo 3: Empacotamento e agregação de objectos

Os objectos sobreviventes ao passo 2 devem ser avaliados acerca da possibilidade de formarem **agregações** (relação forte) ou **pacotes** (relação fraca) de objectos **semanticamente coerentes**.

Exemplo:

O objecto "ficha de cliente" poderá ser empacotado com o objeto "compras de cliente"

4SRS

Passo 4: Associação de objectos

Os objectos devem ser **associados**.

A associação pode ser inferida pela classificação no micropasso 2i ou pelas descrições feitas no micropasso 2iv.

Exemplo:

Um caso de uso classificado como di no micropasso 2i, deverá ter os respectivos objectos (sobreviventes ao passo 2) associados, i.e. uc2.1.2.i deve estar associado a uc2.1.2.d

4SRS

Modelo de Arquitectura Lógica de Software

UML/diagrama de objectos