

41951- ANÁLISE DE SISTEMAS

Determinação de requisitos e OpenUP

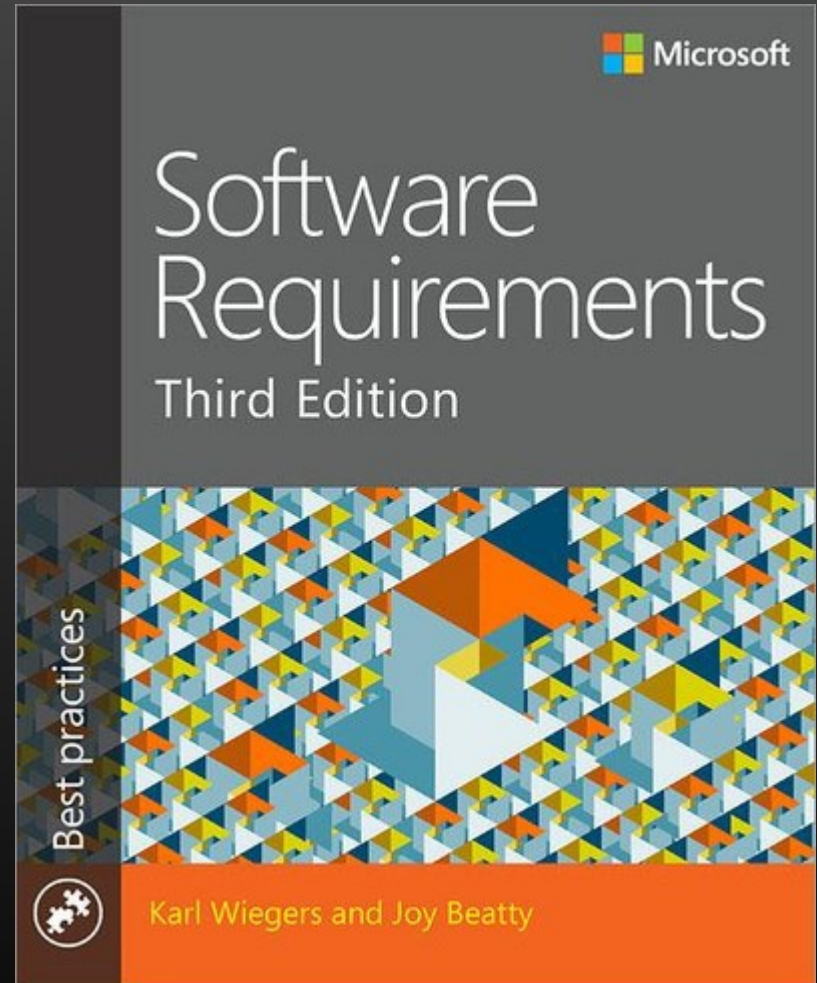
Ilídio | v2022/04/19

Objetivos de aprendizagem

- Distinguir requisitos funcionais e não funcionais
- Enumerar técnicas de recolha de requisitos e argumentar quando usar cada uma
- Descrever técnicas de documentação de requisitos
- Explicar a relação entre requisitos e casos de utilização
- Identificar as atividades e disciplinas relacionadas com os requisitos no OpenUP

Requisitos...

Como descobrir o que é para construir?

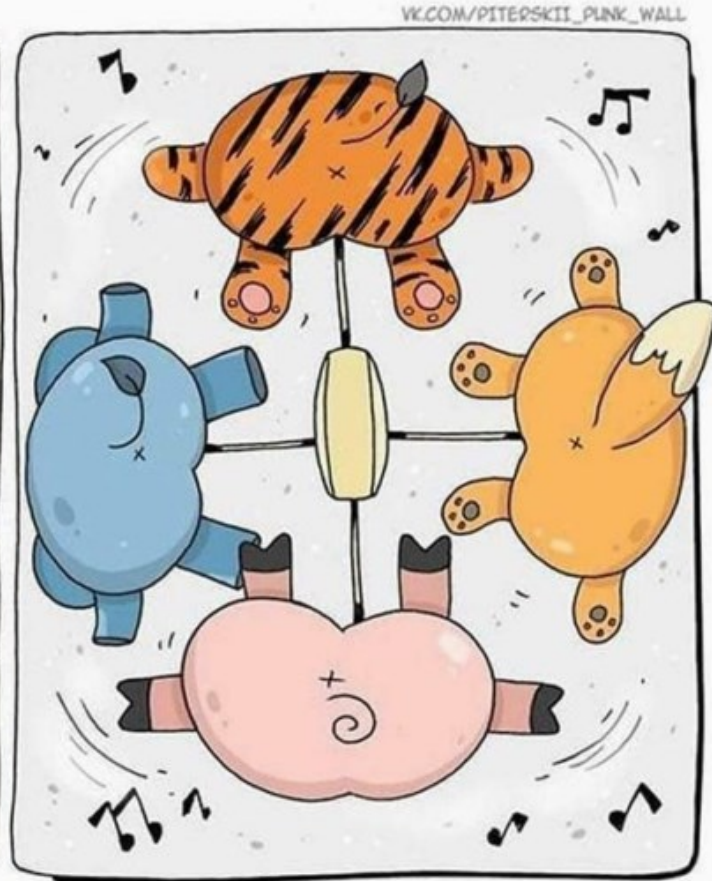


<https://learning.oreilly.com/library/view/software-requirements/9780735679658/>

Clients

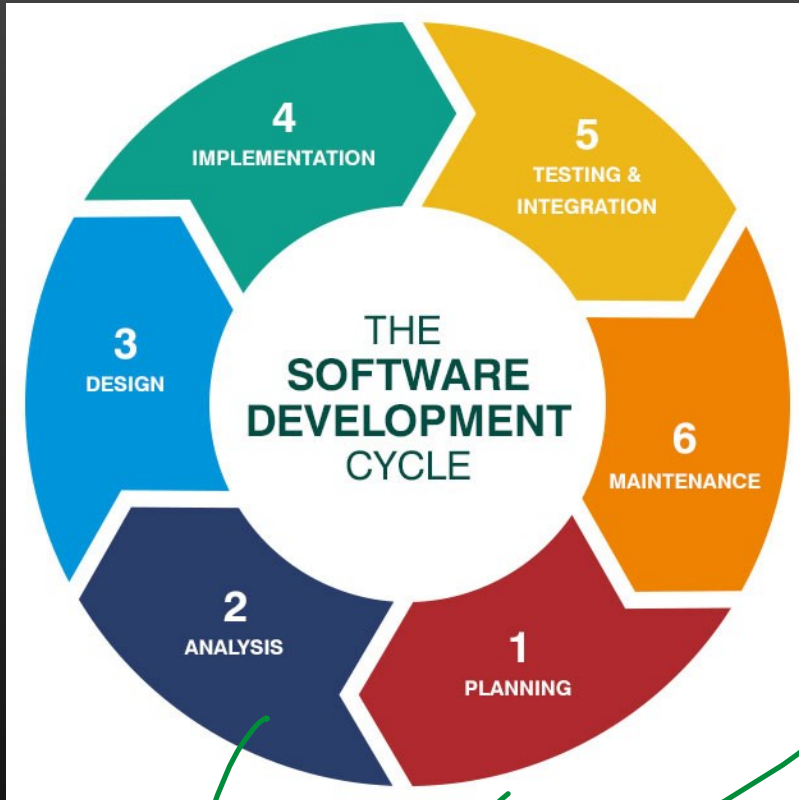


Users

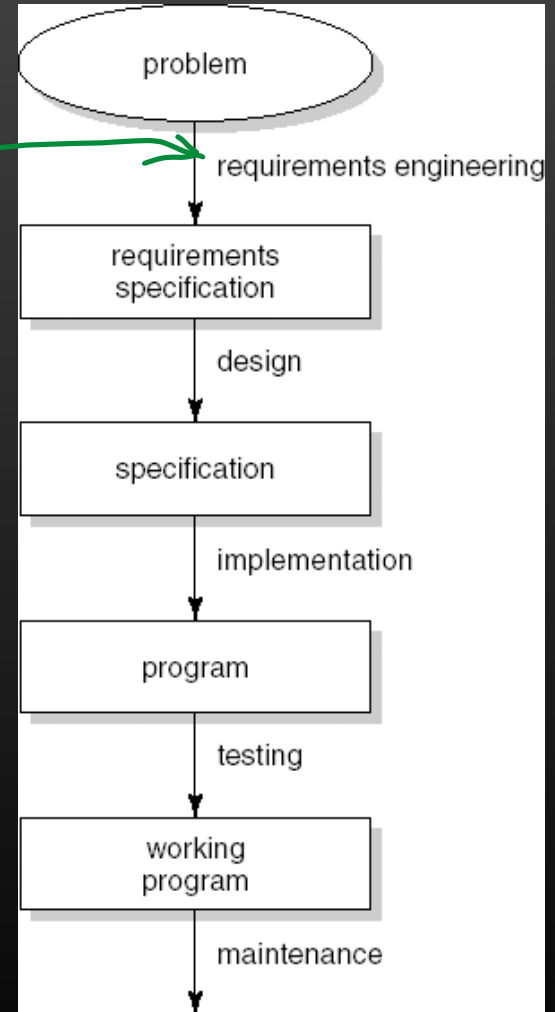


<https://www.designershumor.com/2019/05/21/ux-is-important/>

Atividades e resultados no ciclo de vida do software



Representação do SDLC, com explicitação de QA (#5) e Manutenção (#6).

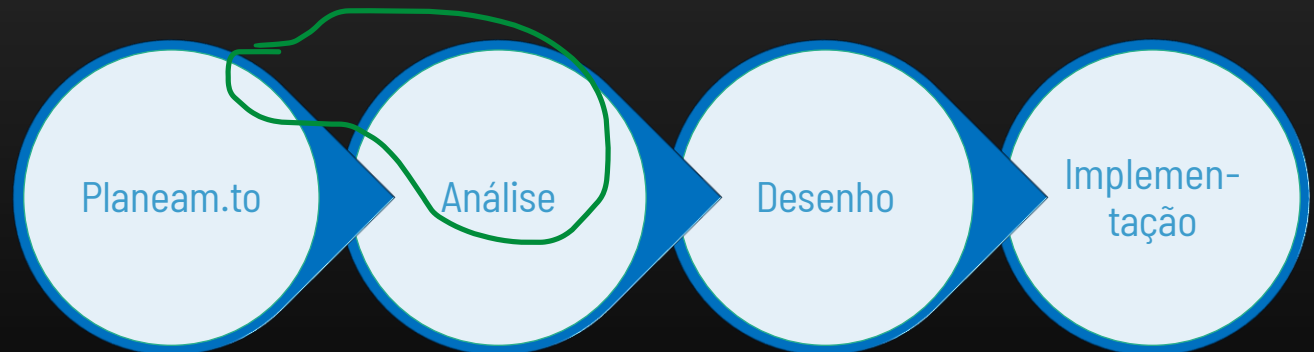


Determinação dos requisitos

Um passo crítico no SDLC

As alterações podem ser feitas facilmente nesta fase

A maioria (>50%) das falhas do sistema são devidas a problemas com os requisitos



Adapted from: Dennis et al, "Systems Analysis and Design: An Object-Oriented Approach with UML", 5th ed.

Porquê realizar actividades de "determinação de requisitos"?

Objetivo

Converter pedidos do negócio (*alto nível*) em requisitos detalhados que possam ser utilizados como inputs para a criação de modelos

O que é um requisito?

Uma declaração do que o sistema deve fazer ou uma característica que deve ter.

evoluirá mais tarde para uma descrição técnica de como o sistema será implementado.

“Missão” do projecto

Desenvolver um sistema que esteja em conformidade com os requisitos estabelecidos (capacidades e condições de operação)

Adapted from: Dennis et al, *“Systems Analysis and Design: An Object-Oriented Approach with UML”*, 5th ed.

Requisitos são muitas vezes elencados em listas “o sistema deve...”

#	Requisito
RF.1	O sistema deve permitir a um profissional criar um novo pedido de adesão , em auto-serviço, na web.
RF.2	O sistema deve gerar credenciais temporárias para os pedidos de adesão e enviá-las, por email, aos solicitantes.
RF.3	O sistema deve permitir a pesquisa de cheques-dentista (emitidos) por número de utente do SNS.
RNF.1	As pesquisas de cheques-dentista têm de retornar resultados em <5 segundos ou um evento de tempo expirado.
...	

Os atributos de qualidade são necessários na definição do produto



As mesmas capacidades funcionais, mas atributos de qualidade diferentes

I Oliveira



Quais são os tipos de requisitos?

Requisitos funcionais

Funcional: refere-se a comportamento → um processo computacional ou tratamento de dados (*as coisas/tarefas que o sistema é capaz de realizar*)

E.g.:

- O sistema deve permitir pesquisar os pacientes pelo nome, usando pelo menos duas palavras (que devem ocorrer em sequência, em qualquer parte do nome).

Requisitos não funcionais

Não-funcional: diz respeito a uma qualidade ou propriedade do sistema (*quão bem deve o sistema fazer a operação*)

E.g.:

- As pesquisas de utentes, por nome, devem retornar os resultados em <1 seg.

A formulação "The system shall..." é comum nos RF.

Exemplo de Definição de Requisitos

Categorias comuns:

- Usabilidade
- Desempenho
- Segurança
- Restrições operacionais
- ...

Adapted from: Dennis et al,
"Systems Analysis and Design:
An Object Oriented Approach
with UML", 5th ed.

Nonfunctional Requirements

1. Operational Requirements

- 1.1. The system will operate in Windows environment.
- 1.2. The system should be able to connect to printers wirelessly.
- 1.3. The system should automatically back up at the end of each day.

2. Performance Requirements

- 2.1. The system will store a new appointment in 2 seconds or less.
- 2.2. The system will retrieve the daily appointment schedule in 2 seconds or less.

3. Security Requirements

- 3.1. Only doctors can set their availability.
- 3.2. Only a manager can produce a schedule.

4. Cultural and Political Requirements

- 4.1. No special cultural and political requirements are anticipated.

Functional Requirements

1. Manage Appointments

- 1.1. Patient makes new appointment.
- 1.2. Patient changes appointment.
- 1.3. Patient cancels appointment.

2. Produce Schedule

- 2.1. Office Manager checks daily schedule.
- 2.2. Office Manager prints daily schedule.

3. Record Doctor Availability

- 3.1. Doctor updates schedule

Agrupamento do RF
depende do problema
específico

O que pode estar envolvido no "desempenho" (atributo de qualidade)?

TABLE 14-2 Some aspects of performance

Performance dimension	Example
Response time	Number of seconds to display a webpage
Throughput	Credit card transactions processed per second
Data capacity	Maximum number of records stored in a database
Dynamic capacity	Maximum number of concurrent users of a social media website
Predictability in real-time systems	Hard timing requirements for an airplane's flight-control system
Latency	Time delays in music recording and production software
Behavior in degraded modes or overloaded conditions	A natural disaster leads to a massive number of emergency telephone system calls

Credit: Wiegiers '13

Os atributos de qualidade do software, segundo Wieger (*-ies)

TABLE 14-1 Some software quality attributes

Analista/requisitos

External quality	Brief description
Availability	The extent to which the system's services are available when and where they are needed
Installability	How easy it is to correctly install, uninstall, and reinstall the application
Integrity	The extent to which the system protects against data inaccuracy and loss
Interoperability	How easily the system can interconnect and exchange data with other systems or components
Performance	How quickly and predictably the system responds to user inputs or other events
Reliability	How long the system runs before experiencing a failure
Robustness	How well the system responds to unexpected operating conditions
Safety	How well the system protects against injury or damage
Security	How well the system protects against unauthorized access to the application and its data
Usability	How easy it is for people to learn, remember, and use the system
Internal quality	Brief description
Efficiency	How efficiently the system uses computer resources
Modifiability	How easy it is to maintain, change, enhance, and restructure the system
Portability	How easily the system can be made to work in other operating environments
Reusability	To what extent components can be used in other systems
Scalability	How easily the system can grow to handle more users, transactions, servers, or other extensions
Verifiability	How readily developers and testers can confirm that the software was implemented correctly

Apesar da implementação da segurança ter aspectos funcionais, é comum classificar os requisitos de segurança como RNF

Credit: Wiegers '13

Funcional ou não-funcional: como distinguir?

Requisitos funcionais

captam o comportamento pretendido

- Serviços e funções que o sistema deverá desempenhar

Podem ser encontrados em descrições de cenários de interação (e.g.: use cases)

Podem ser complementados com diagramas de comportamento: actividades, sequência,...

Requisitos não-funcionais

Restrições globais/qualidades operacionais no software

- E.g.: facilidade de utilização, portabilidade, robustez,...

a.k.a. atributos de qualidade, uma vez que se referem ao grau que se espera de uma certa qualidade pretendida

Normalmente, não se limitam a uma função/módulo, mas sim a uma característica transversal

FURPS: categorias "clássicas" para os requisitos

Functionality

- avaliado através da apreciação do conjunto de capacidades funcionais do programa.

Usability

- avaliados considerando factores humanos, estética geral, consistência, e documentação.

Reliability

- avaliado através da medição da frequência e gravidade das falhas, da exactidão dos resultados de saída, do tempo médio até à falha (MTTF), da capacidade de recuperação das falhas

Performance

- medida utilizando velocidade de processamento, tempo de resposta, consumo de recursos, rendimento e eficiência.

Supportability

- combina capacidade de extensão, capacidade de **adaptação e de manutenção** e, além disso, facilidade de teste, compatibilidade, facilidade de configuração, facilidade de instalação de um sistema e facilidade de localização de problemas.

Requirements

STRQ1: Want to be able to transfer funds from other accounts (not necessarily held with this firm) to a trading account.

STRQ2: State and federal regulations require monthly reports of account activity. Refer to specification RUFS-1234 for details of the information required.

STRQ3: The system should allow the use of any browser.

STRQ4: Customers want to manage their retirement funds.

STRQ5: Must be able to upgrade the system without taking it offline.

STRQ6: The system should allow traders to trade in multiple markets across the world.

STRQ7: Must be able to provide convenient answers to customer's most common questions.

STRQ8: The system must provide a secure environment that prohibits fraudulent access.

STRQ9: Need a way to train customers in the use of the system quickly and conveniently.

STRQ10: The system must operate on hardware that falls under the company's current maintenance contracts.

STRQ11: Need to be able to maintain the system with our current IT hardware and skills. Refer to enterprise architecture document EA-1234 for details.

STRQ12: Need account activity statements for tax reporting.

STRQ13: The system must provide all the basic capabilities of a normal stock broking firm.

STRQ14: Need to be able to perform research on any given stock.

STRQ15: The system must allow traders to obtain up-to-date news and alerts on nominated stock.

STRQ16: The system must provide current and historical information on Trading Accounts. Such as number of shares held, current price, total Trading Account value

STRQ17: The system shall provide the following types of trades: Market Trades (buy and sell), Limit Trades (buy and sell), and transfers between mutual funds.



Software designers tend to focus on the problem to be solved. Just don't forget that the FURPS attributes are always part of the problem. They must be considered.

A importância relativa dos diferentes atributos de qualidade depende do projeto

Loja on-line → muitas sessões em simultâneo, transações seguras,...

App Dialer (telefonar) → uso intuitivo, comunicação clara do estado da chamada

Homebanking → segurança, disponibilidade,...

Explorar:

- Wiegers, chap. 14

Regra do negócio (“business rule”)

Business rule

→ uma política, orientação, norma, ou regulamento que define ou restringe algum dos aspectos do negócio.

Regra vs Requisito

Não um requisito de software em si (porque têm uma existência para além dos limites de qualquer aplicação de software específica), mas é a origem de vários tipos de requisitos de software.

→ Mais info...

Exemplos:

- "Um novo cliente deve pagar 30% da taxa de consultoria estimada e das despesas de viagem, em adiantamento".
- "Os clientes devem ser maiores de idade (≥ 18 anos)."
- "Há uma joia de admissão, a pagar na inscrição, no valor de 50% da mensalidade."

Estas características *não dependem do software A ou B*, mas terão de ser tidas em consideração na implementação do software.

Técnicas de Recolha de Requisitos

Processo utilizado para :

Descobrir todos os requisitos (os descobertos tardiamente são mais difíceis de incorporar)

Obter apoio e confiança entre os utilizadores

Qual(is) técnica(s) a utilizar?

1. Entrevistas
2. Workshops de Desenho Colaborativo (JAD)
3. Questionários Análise de documentos +
4. Observação
5. Focus Group

Credit: Dennis et al, *"Systems Analysis and Design: An Object Oriented Approach with UML"*, 5th ed.


I Oliveira

OpenUP as práticas de requisitos

OpenUP practices

[OpenUP](#) > Practices >
Technical Practices >
Shared Vision >
Requirements Gathering
Techniques

Detalhes disponíveis na
documentação do
OpenUp



OpenUP

Requirements Gathering Techniques

After you have identified these sources, there are a number of techniques following will describe the various techniques, followed by a brief discussion.

To get the requirements down on paper, you can do one or more of the following:

- Conduct a brainstorming session
- Interview users
- Send questionnaires
- Work in the target environment
- Study analogous systems
- Examine suggestions and problem reports
- Talk to support teams
- Study improvements made by users
- Look at unintended uses
- Conduct workshops
- Demonstrate prototypes to stakeholders

The best idea is to get the requirements down quickly and then to encourage those corrections, and repeat the cycle. Do it now, keep it small, and you can devise, but expect to keep on correcting it throughout the process. correct it immediately.

Técnicas de recolha de requisitos comparadas

Pode ser usada uma combinação de técnicas

A análise e observação de documentos requer pouca formação; as sessões JAD podem ser muito desafiantes

	Interviews	Joint Application Design	Questionnaires	Document Analysis	Observation
Type of information	As-is, improvements, to-be	As-is, improvements, to-be	As-is, improvements	As-is	As-is
Depth of information	High	High	Medium	Low	Low
Breadth of information	Low	Medium	High	High	Low
Integration of information	Low	High	Low	Low	Low
User involvement	Medium	High	Low	Low	Low
Cost	Medium	Low-Medium	Low	Low	Low to Medium

Credit: Dennis et al, "Systems Analysis and Design: An Object Oriented Approach with UML", 5th ed.

Figure 7-3 suggests the elicitation techniques that are most likely to be useful for various types of projects. Select the row or rows that represent characteristics of your project and read to the right to see which elicitation techniques are most likely to be helpful (marked with an X). For instance, if you're developing a new application, you're likely to get the best results with a combination of stakeholder interviews, workshops, and system interface analysis. Most projects can make use of interviews and workshops. Focus groups are more appropriate than workshops for mass-market software because you have a large external user base but limited access to representatives. These suggestions for elicitation techniques are just that—suggestions. For instance, you might conclude that you do want to apply user interface analysis on mass-market software projects.

	Interviews	Workshops	Focus groups	Observations	Questionnaires	System interface analysis	User interface analysis	Document analysis
Mass-market software	x		x		x			
Internal corporate software	x	x	x	x		x		x
Replacing existing system	x	x		x		x	x	x
Enhancing existing system	x	x				x	x	x
New application	x	x				x		
Packaged software implementation	x	x		x		x		x
Embedded systems	x	x				x		x
Geographically distributed stakeholders	x	x			x			

FIGURE 7-3 Suggested elicitation techniques by project characteristic.

Técnicas alternativas

Análise da área (domínio)

Estudar as características do domínio

Aprender com os outros

Aumentar a propensão para escalar para mais clientes

Mapas de conceitos

Representam relações importantes entre conceitos

Concentrar as pessoas num pequeno número de ideias-chave

User Stories

Associado a métodos de desenvolvimento ágeis

Pouca tecnologia necessária, facilmente actualizável

→ A explorar mais à frente no semestre

“lista de pedidos para a casa...”



Requirements elicitation (Determinação dos requisitos)

Sistema a construir

Requisitos documentados na análise

Inconsistente

Inexequível
(tecnologia)

Requisitos
em falta

supérfluo (desnecessário)

Âmbito inicial

© Oliveira

Quais são as actividades de determinação dos requisitos?

O núcleo do desenvolvimento de requisitos é a elicitação

o processo de identificação das necessidades e restrições dos vários intervenientes para um sistema de software.

Elicitação não é a mesma coisa que "recolha de requisitos". Nem é uma simples questão de transcrever exactamente o que os utilizadores dizem.

Elicitação é um processo colaborativo e analítico que inclui actividades para recolher, descobrir, extrair e definir requisitos.

A elicitação é utilizada para descobrir requisitos focados no negócio + focados no utilizador + funcionais + não-funcionais.

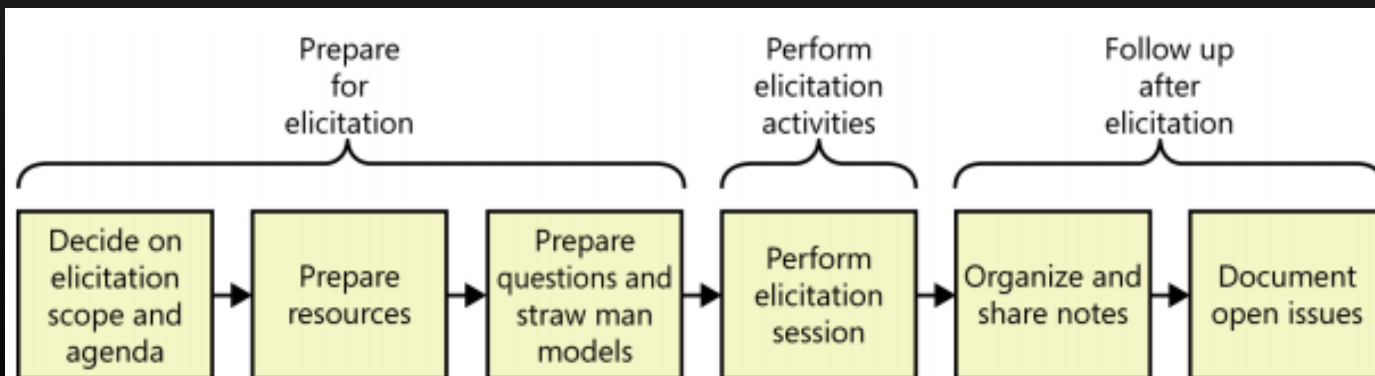


FIGURE 7-2 Activities for a single requirements elicitation session.

elicitação de requisitos como um processo

A elicitação de requisitos é talvez o aspecto mais desafiante, crítico, propenso a erros, e de comunicação intensiva no desenvolvimento de software.



Documentação dos requisitos

Requisitos são muitas vezes elencados em listas “o sistema deve...”

#	Requisito
RF.1	O sistema deve permitir a um profissional criar um novo pedido de adesão , em auto-serviço, na web.
RF.2	O sistema deve enviar credenciais temporárias para os pedidos de adesão e enviá-las, por email, aos solicitantes, até 10min depois do registro.
RF.3	O sistema deve permitir a pesquisa de cheques-dentista (emitidos) por número de utente do SNS.
RNF.1	As pesquisas de cheques-dentista têm de retornar resultados em <5 segundos ou um evento de tempo expirado.
...	

"The system shall be 100% reliable and 100% available".

"The system shall have a minimum response to a query of 1 second irrespective of system load".



AVL-1. The system shall be at least 95 percent available on weekdays between 6:00 A.M. and midnight Eastern Time, and at least 99 percent available on weekdays between 3:00 P.M. and 5:00 P.M. Eastern Time.

IOP-1. The Chemical Tracking System shall be able to import any valid chemical structure from the ChemDraw (version 13.0 or earlier) and MarvinSketch (version 5.0 or earlier) tools.

PER-1. Authorization of an ATM withdrawal request shall take no more than 2.0 seconds.

PER-2. The anti-lock braking system speed sensors shall report wheel speeds every 2 milliseconds with a variation not to exceed 0.1 millisecond.

PER-3. Webpages shall fully download in an average of 3 seconds or less over a 30 megabits/second Internet connection.

PER-4. At least 98 percent of the time, the trading system shall update the transaction status display within 1 second after the completion of each trade.

S. M. A. R. T. → Specific, Measurable, Attainable, Relevant and time-sensitive (*Específico, Mensurável, Atingível, Relevante e Rastreável no tempo*)

Step 5: Specify well-structured quality requirements

Simplistic quality requirements such as “The system shall be user-friendly” or “The system shall be available 24x7” aren’t useful. The former is far **too subjective and vague**; the latter is **rarely realistic or necessary**. **Neither is measurable**. Such requirements provide little guidance to developers. So the final step is to **craft specific and verifiable requirements** from the information that was elicited regarding each quality attribute. When writing quality requirements, keep in mind the useful SMART mnemonic—make them *Specific, Measurable, Attainable, Relevant, and Time-sensitive*.

Credit: Wiegiers '13

O que são requisitos bem formulados? (ISO-IEEE 29148)

Uma formulação de uma característica que:

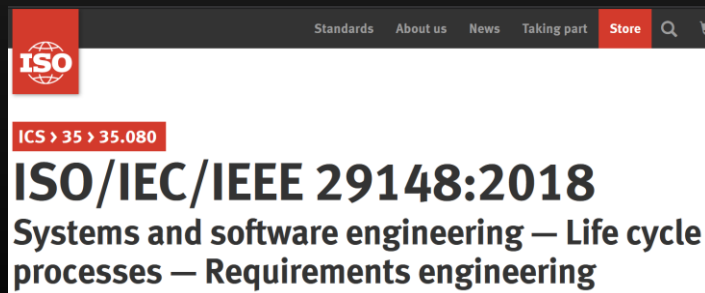
- tem de ser satisfeita ou detida por um sistema para resolver um problema ou para atingir um objectivo de alguma "parte interessada"
- pode ser verificada,
- é qualificado por condições mensuráveis e limitado por restrições,
- e define o comportamento ou a capacidade do sistema quando usado, mas não uma capacidade do utilizador ou operador.

Elementos de estilo

Um requisito é uma declaração que expressa uma necessidade e as limitações e condições que lhe estão associadas.

Se expressa em linguagem natural, a declaração deve incluir um sujeito, um verbo e um complemento. Um requisito deve indicar o sujeito do requisito (por exemplo, o sistema, o software, etc.) e o que deve ser feito (por exemplo, operar a um nível de potência).

E.g: O Sistema de Facturação [Sujeito],
deve exibir as facturas pendentes do cliente [Acção] por ordem crescente
[Complemento] em que as facturas devem ser pagas.



Requisitos (e casos de utilização) no OpenUP

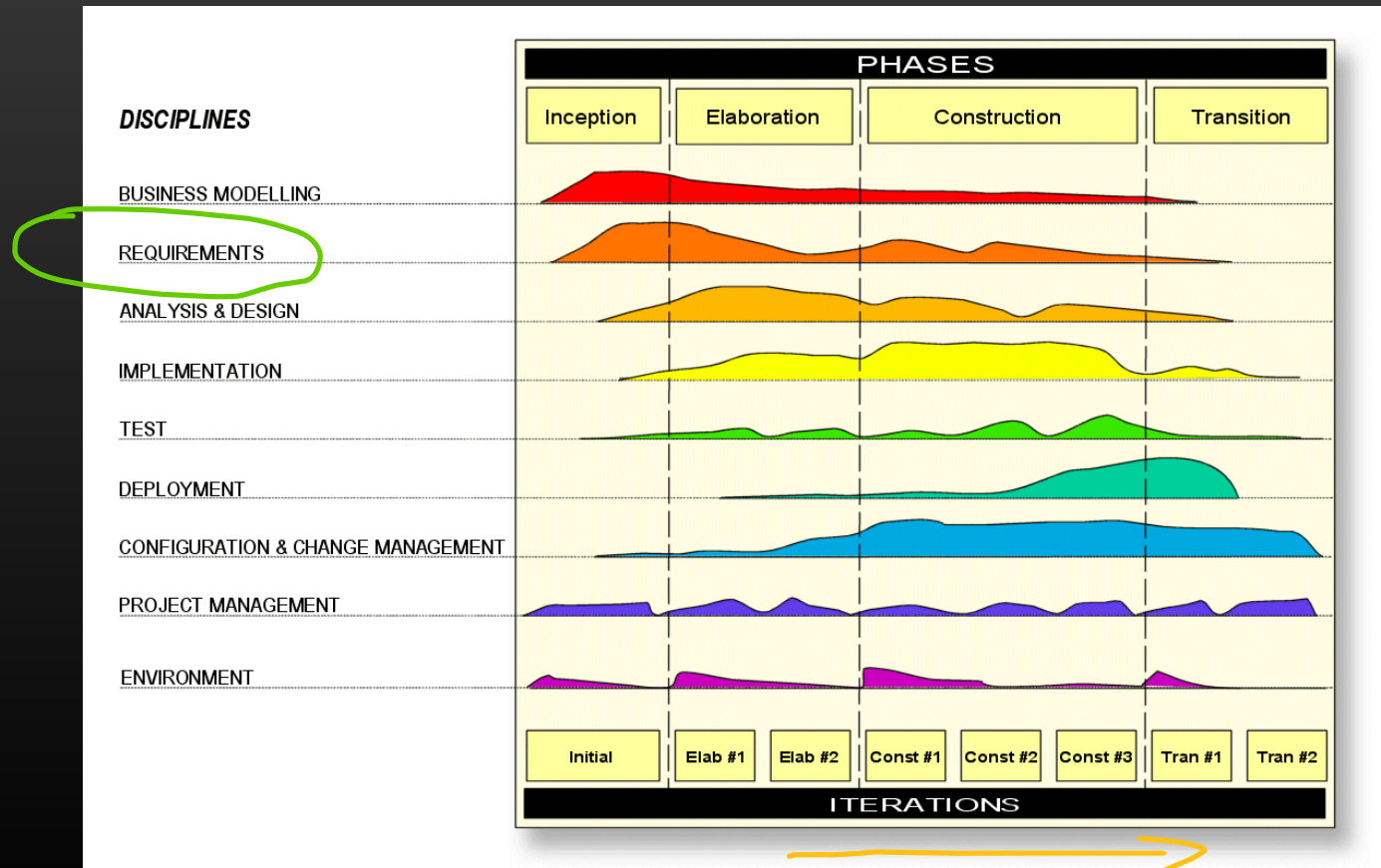
Visão geral do OpenUP/Unified Process

Os requisitos não são estanques; podem ser evoluídos → refinados

O UP oferece uma abordagem ao SDLC concebida como uma matriz, cruzando diferentes disciplinas técnicas com iterações (evoluções) no projecto.

(Nota: fases UP ≠ fases SDLC)

A análise dos requisitos é realizada principalmente no início do projeto (requisitos básicos), mas também durante as iterações (requisitos evolutivos).



Which main elicitation approaches exist?

What is the goal the **user** wants to achieve? vs.

What capability should the **product/system** possess?



OpenUP: como identificar e descrever os requisitos, para que o âmbito fique determinado

The screenshot shows the OpenUP web application interface. At the top, there is a header with the OpenUP logo and navigation links for Glossary, Feedback, and About. Below the header, a breadcrumb trail indicates the current location: Practices > Technical Practices > Use Case Driven Development > Tasks > Identify and Outline Requirements. The left sidebar contains a 'Where am I' section with a 'Tree Sets' button and a 'Team' dropdown. Below this is a tree view of the application structure, including Introduction to OpenUP, Getting Started, Delivery Processes, Practices (expanded), Management Practices, Technical Practices (expanded), Concurrent Testing, Continuous Integration, Evolutionary Architecture, Evolutionary Design, Shared Vision, Test Driven Development, Use Case Driven Development (expanded), How to Adopt the Use Case Driven Development, and Key Concepts. The main content area displays the 'Task: Identify and Outline Requirements' page. It features a yellow arrow icon and a description: 'This task describes how to identify and outline the requirements for the system so that the scope of work can be determined.' Below this, it lists 'Disciplines: Requirements'. There are buttons for 'Expand All Sections' and 'Collapse All Sections'. A 'Purpose' section is highlighted with a blue header, containing a detailed description of the task's purpose. At the bottom right of the main content area, there is a 'Back to top' link.

OpenUP

Glossary | Feedback | About

Print

Where am I | Tree Sets |

Team

- Introduction to OpenUP
- Getting Started
- Delivery Processes
- Practices
 - Management Practices
 - Technical Practices
 - Concurrent Testing
 - Continuous Integration
 - Evolutionary Architecture
 - Evolutionary Design
 - Shared Vision
 - Test Driven Development
 - Use Case Driven Development
 - How to Adopt the Use Case Driven Development
 - Key Concepts

Practices > Technical Practices > Use Case Driven Development > Tasks > Identify and Outline Requirements

Task: Identify and Outline Requirements

This task describes how to identify and outline the requirements for the system so that the scope of work can be determined.

Disciplines: [Requirements](#)

[Expand All Sections](#) [Collapse All Sections](#)

Purpose

The purpose of this task is to identify and capture functional and non-functional requirements for the system. These requirements form the basis of communication and agreement between the stakeholders and the development team on what the system must do to satisfy stakeholder needs. The goal is to understand the requirements at a high-level so that the initial scope of work can be determined. Further analysis will be performed to detail these requirements prior to implementation.

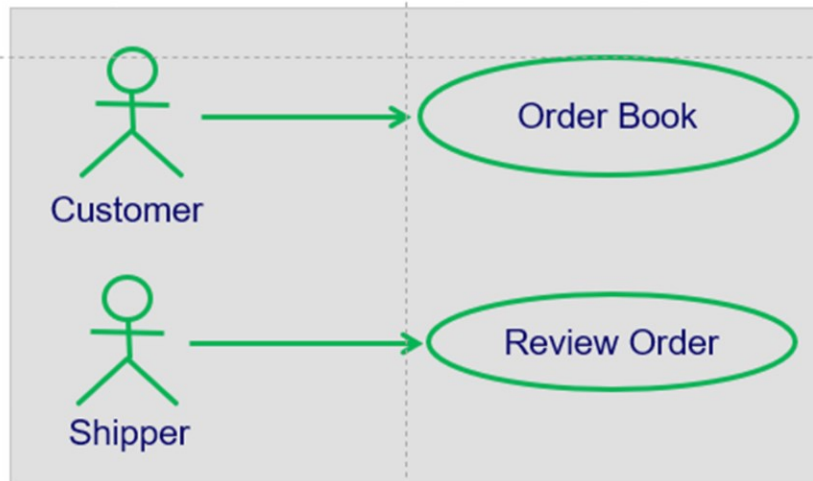
[Back to top](#)

OpenUP recommended practices

Use Case Driven Development 🏆



- This practice describes how to capture requirements with a combination of use cases and system-wide requirements, and then drive development and testing from those use cases.



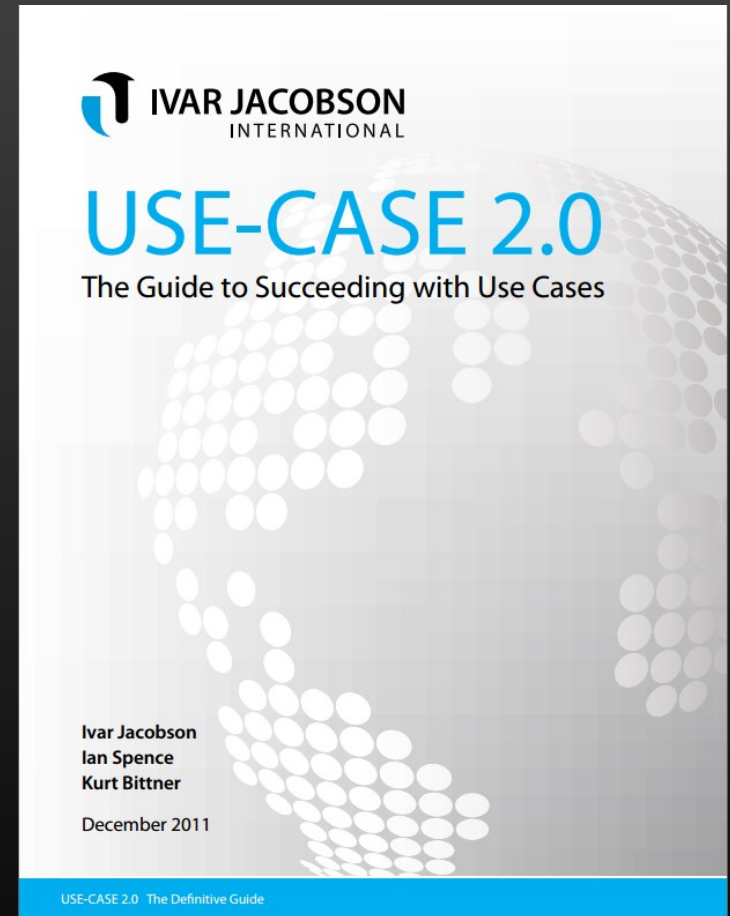
Caso de utilização (CaU)

Engloba uma sequência de ações que um sistema executa e que produzem um resultado com valor para algum ator em particular.

Implica:

- Foco no utilizador do sistema e nos episódios de uso
- Foco na compreensão daquilo que os atores consideram um resultado relevante (motivações para usar um sistema → problemas que querem resolver)

Conceito apresentado originalmente em:
Jacobson, I., 1993. *Object-oriented software engineering: a use case driven approach*.
Pearson Education.



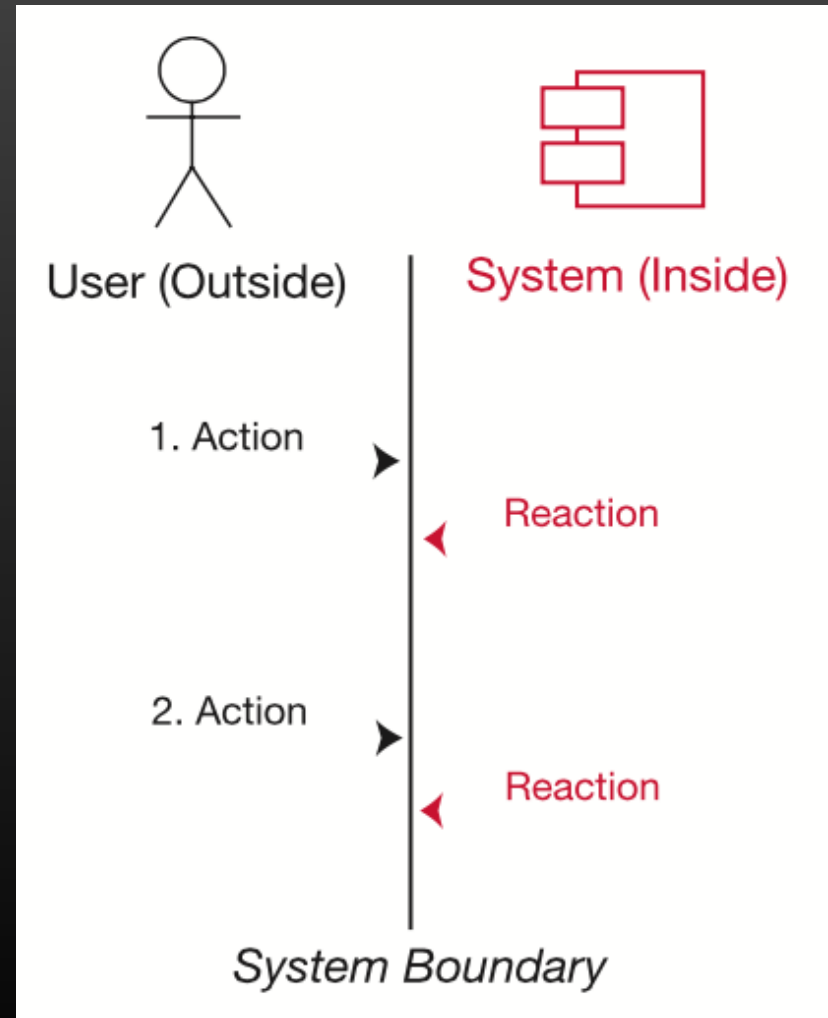
<https://www.ivarjacobson.com/publications/white-papers/use-case-ebook>

O CaU descreve **um diálogo entre o ator e o sistema**

Narrativas para contar como é que “alguém” usa um sistema

CaU: Comprar produtos (supermercado)

1. Um Cliente chega a uma caixa com artigos para comprar.
2. O Operador passa cada artigo no leitor de código de barras para registo.
3. O sistema apresenta o total provisório e a lista de artigos incluídos.
4. O Operador termina a venda e indica o tipo de pagamento.
5. O cliente introduz a informação de pagamento.
6. O sistema valida o pagamento, atualiza o stock e imprime o recibo.
7. O cliente leva o recibo e os artigos.



Use case:	Brief description:
Create new assignment	The Teaching Staff creates a new Activity of type Assignment, directly inserting it in the page layout. The assignment must define a title and a time period for submissions and can be configured to work with individual or group submissions. The assignment is listed in the student view and on the specified date (or immediately, if none is given) accepts submissions from registered students.

Use case:	<u>Add new assignment</u>
Brief description:	The Faculty creates assignments for students, directly inserting it in the course page. The assignment defines a time period for submissions and can be configured to work with individual or group submissions. The assignment is listed in the student view and on the specified date (or immediately, if none is given) accepts submissions from students.
Basic flow:	<ol style="list-style-type: none"> 1. Log-in using corporate IdP. 2. Select desired course. 3. Turn editing mode on. 4. Add Assignment activity in the page layout. 5. Configure Assignment activity. 6. Commit changes.
Alternative flows:	<p>Step 1: IdP unavailable.</p> <p>Step 4/5: Instead of a new, empty assignment, the user may reuse an existing one.</p>
Open issues:	<p>Step 3/4. The course is closed. Are changes allowed to past courses?</p> <p>Step 5. The browser does not accept the rich text editor. Default to plain text?</p>

Conclusões...

Outside the Developing Organization

Direct user	Business management	Consultant
Indirect user	Contracting officer	Compliance auditor
Acquirer	Government agency	Certifier
Procurement staff	Subject matter expert	Regulatory body
Legal staff	Program manager	Software supplier
Contractor	Beta tester	Materials supplier
Subcontractor	General public	Venture capitalist

Developing Organization

Development manager	Sales staff	Executive sponsor
Marketing	Installer	PMO
Operational support	Maintainer	Manufacturing
Legal staff	Program manager	Training staff
Information architect	Usability expert	Portfolio architect
Company owner	Shareholder	Infrastructure support

Project Team

Project manager	Tester	Product owner
Business analyst	Product manager	Data modeler
Application architect	QA staff	Process analyst
Designer	Doc writer	Hardware engineer
Developer	DBA	Infrastructure analyst

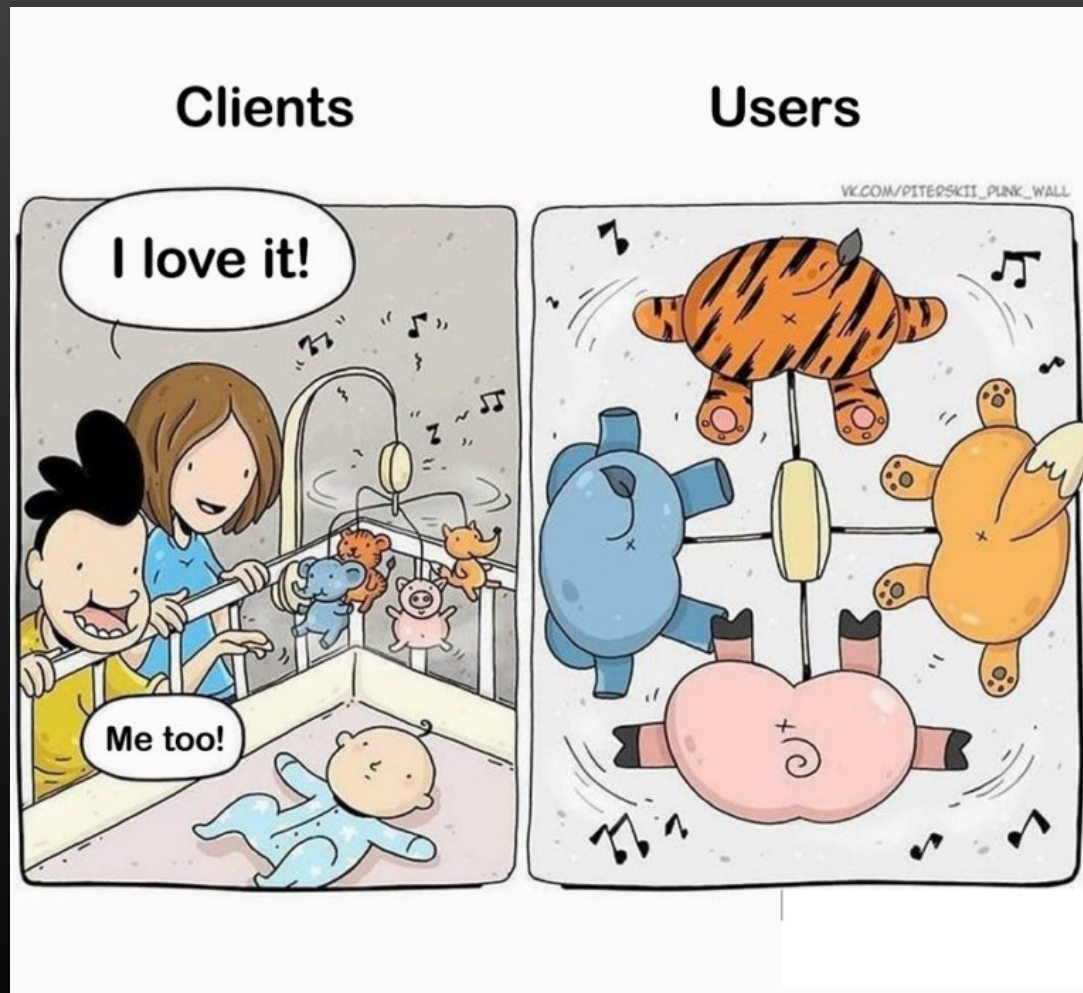
Figure 1. Some possible software project stakeholders.

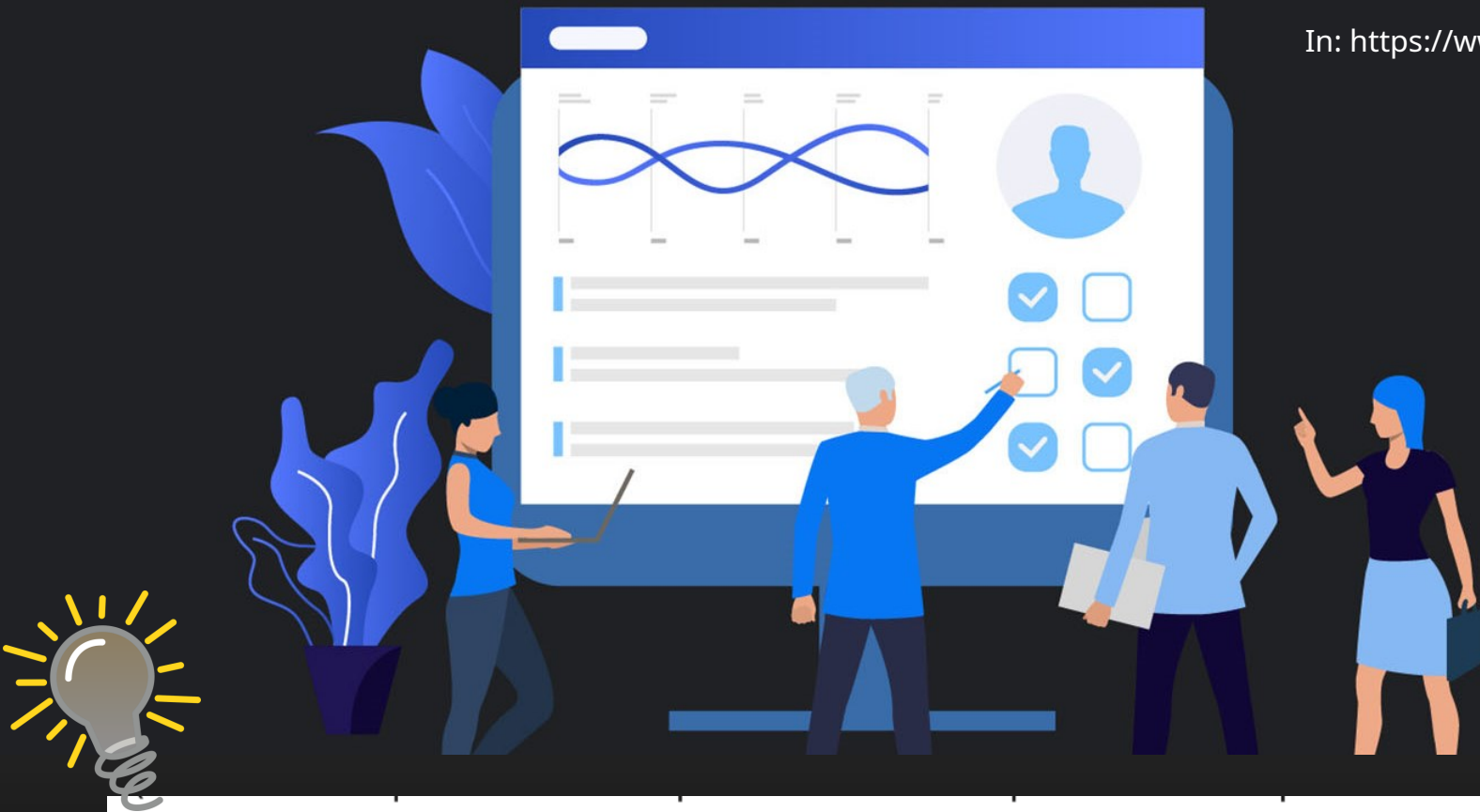
Compromisso...

Business...

Utilizadores...

Viabilidade...









And second, those of us in the software domain tend to be enamored with technical and process solutions to our challenges. We sometimes fail to appreciate that **requirements elicitation**—and much of software and systems project work in general—is **primarily a human interaction challenge**. No magical new techniques have come along to automate that, although various tools are available to help geographically separated people collaborate effectively.

In: Wiegers, “Software Requirements”

Ver também

Karl Wiegiers



Aug 12, 2019 · 10 min read ★ ·

Ten Cosmic Truths About Software Requirements

These facts apply to nearly every project. Ignore them at your peril.



Readings & references

Core readings	Suggested readings
<ul style="list-style-type: none">• [Dennis15] – Chap. 3 – Requirements Determination	<ul style="list-style-type: none">• [Pressman15] – Chap. 8 – Understanding Requirements• [Wiegers13] – Chap. 1 -3