Requisitos de Software

Valores e Princípios da Engenharia de Requisitos

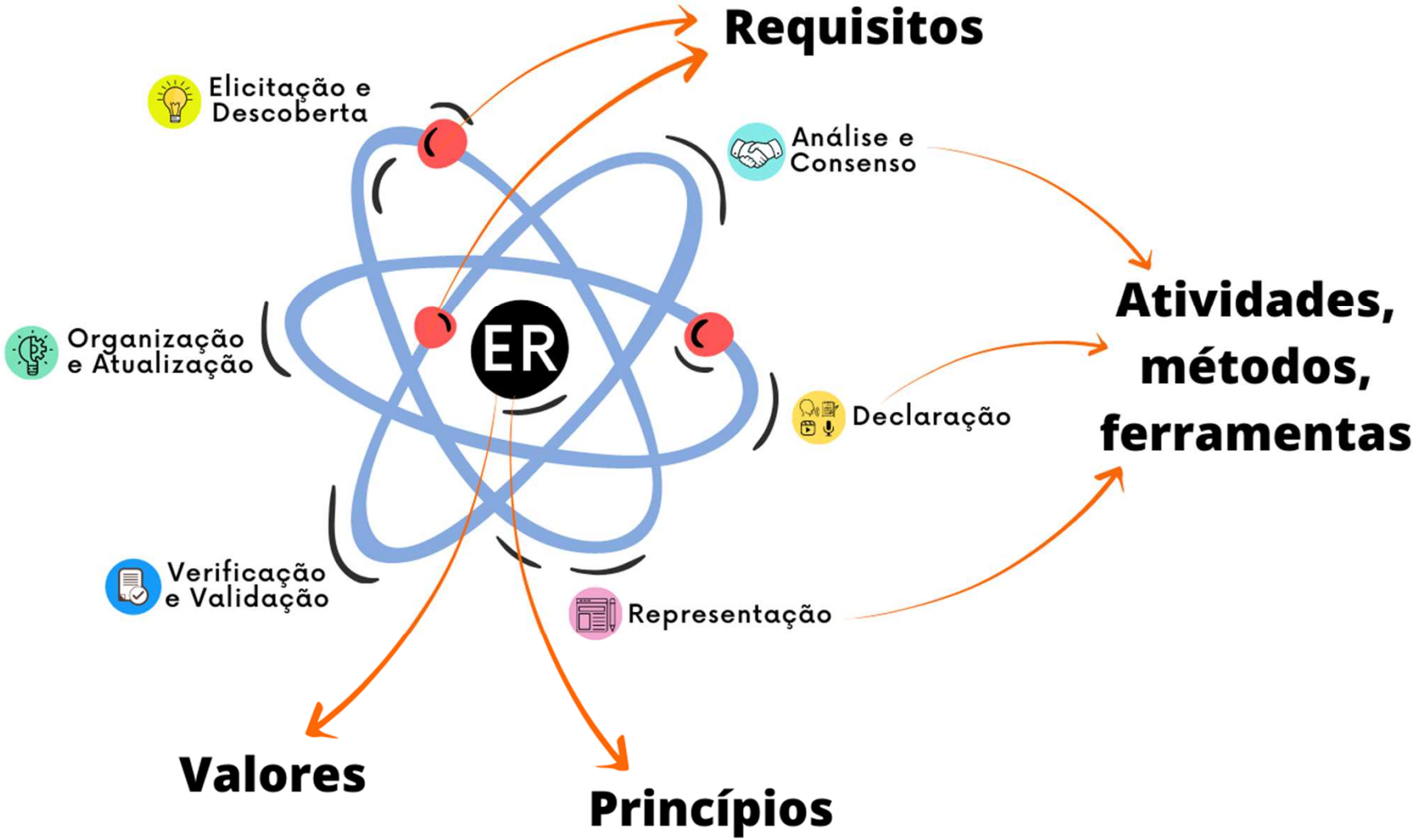
Fonte, baseado em:
Handbook for the CPRE Foundation Level, version 1.1.0, septempber 2022.
Suzanne Robertson and James Robertson, Mastering the Requirements Process, Third Edition, 2014.


georgemarsicano@unb.br


George Marsicano

UnB

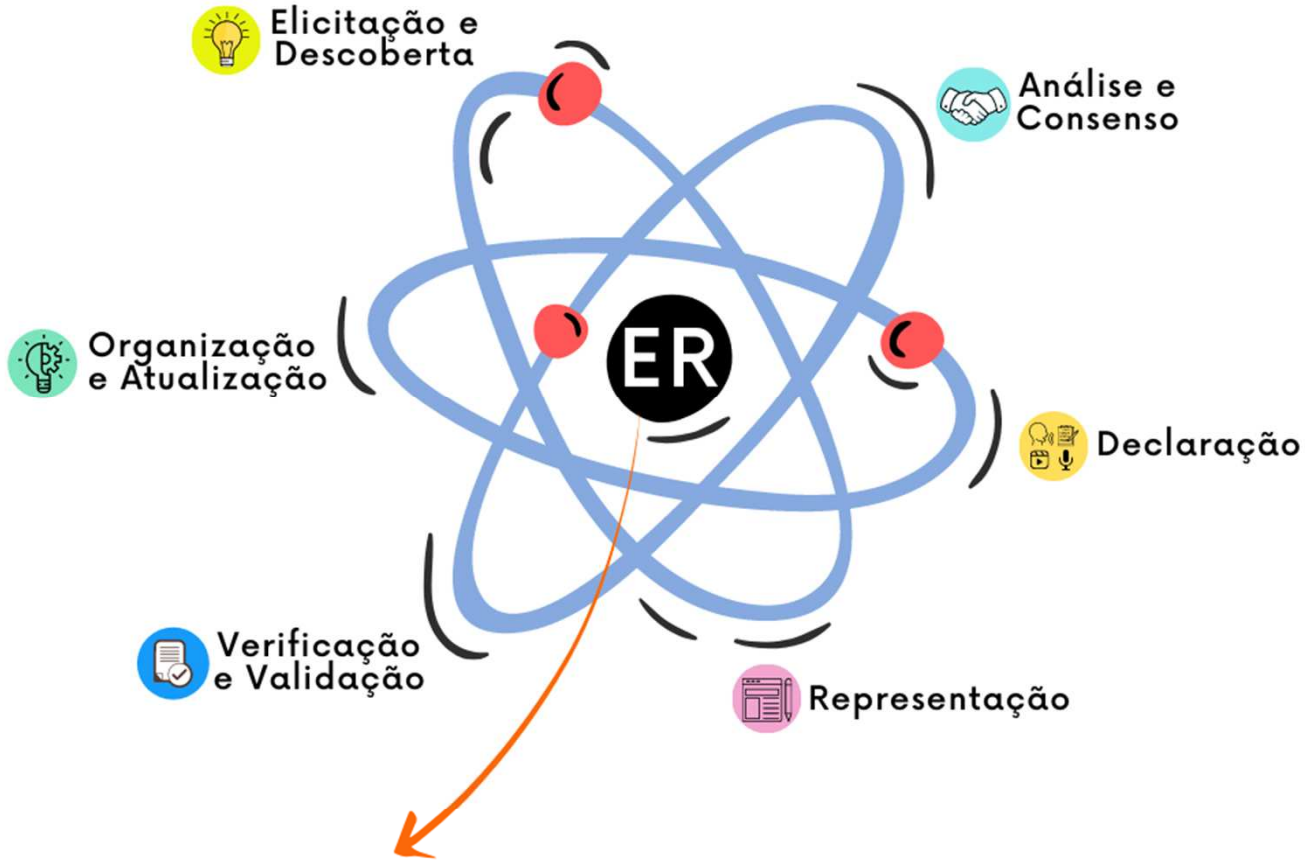
CEDIS Centro de Estudos, Desenvolvimento e Inovação em Software



UnB

CEDIS Centro de Estudos, Desenvolvimento e Inovação em Software

cedis.unb.br



Valores

- Comunicação
- Simplicidade
- Feedback
- Coragem
- Respeito
- Compromisso
- Confiança

UnB

CEDIS Centro de Estudos, Desenvolvimento e Inovação em Software

cedis.unb.br

Os 7 Valores da ER

- Os valores aqui propostos para a ER, são derivados e adaptados do eXtreme Programming (XP) e do Scrum, acrescidos da *confiança*.
- Ressalta-se que, aqui não há a intenção de que a ER seja direcionada a ser ágil ou dirigida a plano, e sim, que seja adaptável ao contexto de aplicação.
- Os valores indicados não são os únicos possíveis, e sim, os *valores de condução da ER*.
- O mais importante é *alinhar o comportamento* da equipe *aos valores da equipe*.

Comunicação

- Ter foco na comunicação é essencial para que seja possível *transmitir os requisitos*, assim como *compartilhar informações* e *conhecimentos*.
- A comunicação *incentiva o feedback*.
- A comunicação é importante para criar um *senso de equipe* e *cooperação* eficaz entre as partes interessadas.
- *Toda a comunicação*, seja ela face-a-face, escrita, em áudio, em vídeo, por meio de representações, etc., deve ser *adequada ao seu contexto de aplicação*, assim como *às pessoas* que irão utilizá-la.

Feedback

- A compreensão das *necessidades dos usuários* e do *negócio* propriamente dito são um *aprendizado constante*.
- As equipes de software devem se esforçar para gerar o *máximo de feedback* possível, o *mais rápido* possível em relação aos requisitos.
- Quanto mais cedo você souber de problemas ou novas necessidades com os requisitos, mais cedo poderá se *adaptar*.
- Estando satisfeitos com a *melhoria e refinamento dos requisitos* em vez de esperar a perfeição instantânea, o feedback deve ser utilizado para aproximar a equipe e o cliente aos objetivos da solução de software.

Feedback (cont.)

- O feedback vem de várias formas, como:
  - Opiniões sobre uma ideia, sua ou de seus colegas de equipe
  - Se a organização do backlog está adequada
  - Se os requisitos estão fáceis de se entender
  - Se o nível de granularidade da declaração dos requisitos é adequado ao tipo de comunicação estabelecida entre os envolvidos (clientes, usuários, equipe de software)
- O feedback é uma parte crítica da *comunicação*.
- O feedback também contribui para a *simplicidade*.

Simplicidade

- A simplicidade é o valor mais *intensamente intelectual*.
- Tornar um requisitos simples o suficiente para *resolver* apenas o *problema de hoje* é um trabalho árduo.
  - *Síndrome de Nostradamus*: mesmo sabendo o que o cliente quer *hoje*, a equipe insiste em tentar desenvolver uma solução que também resolva problemas do *futuro*.
- A simplicidade só faz sentido dentro de um *contexto* (produto, equipe, cliente).
- *Melhorar a comunicação* ajuda a alcançar a simplicidade, *eliminando requisitos desnecessários* ou *adiáveis* das preocupações atuais.

Coragem

- Coragem é *ação efetiva* diante do medo.
- A coragem sozinha é perigosa, mas em conjunto com os outros valores ela é *poderosa*.
- A coragem de falar *verdades*, *agradáveis* ou *desagradáveis*, fomenta a *comunicação* e a *confiança*.
- A coragem de descartar soluções que falham e buscar novas incentiva a *simplicidade*.
- A coragem de buscar *respostas reais* e *concretas* gera *feedback*.

Coragem (cont.)

- A equipe de engenharia de software deve ter coragem para:
  - Ser *protagonista* nas conversas junto ao cliente
  - Ser *responsável* pelos requisitos resultantes da interação com o cliente
  - Buscar *conhecimentos* sobre o *domínio do negócio*
  - Buscar *técnicas* que *facilitem* a *interação* com o cliente
  - *Adaptar-se* as características do cliente, negócio e contexto
  - *Confiar* em suas *práticas*, bem como nos seus *colegas de equipe*
  - Construir *relações de confiança* com os seus *clientes*;

- A *coragem* faz-se ainda mais necessária nos *momentos de crise.*

Coragem (cont.)

- O *cliente* teme:
  - Não obter o que foi solicitado/contratado;
  - Pedir a coisa errada;
  - Não entender o que o engenheiro de software fala ou faz;
  - Não saber o que está acontecendo (falta de feedback);
  - Ater-se às primeiras decisões de projeto e não ser capaz de reagir à mudança do negócio.

Coragem (cont.)

- O *engenheiro de software* teme:
  - Não saber como interagir com o cliente;
  - Identificar o problema incorreto;
  - Não saber elicitar e descobrir os requisitos;
  - Não saber comunicar os requisitos adequadamente;
  - Não saber como garantir que os requisitos consensuadas estejam implementados na solução.

Respeito

- Toda pessoa cuja vida é tocada pelo desenvolvimento de software tem o mesmo *valor como ser humano*.
- *Ninguém vale* intrinsecamente *mais do que ninguém*.
- Para que o desenvolvimento de software melhore simultaneamente em *humanidade* e *produtividade*, as contribuições de cada pessoa da equipe precisam ser respeitadas.
- *Eu* sou importante e *você* também.

Compromisso

- É o resultado de uma conversa que possui o objetivo de alcançar uma compreensão mútua sobre "quem" fará "o que" e "quando".
- Os compromissos são estabelecidos a partir de um pedido/oferta + uma declaração de aceitação.

- Se os membros de uma equipe não estão comprometidos:
  - Uns com os outros e com o que estão fazendo, ao *ER não funcionará*.
  - Com um projeto, *nada pode salvá-lo*.
  - Com os clientes, *nenhuma interação resolverá*.

Confiança

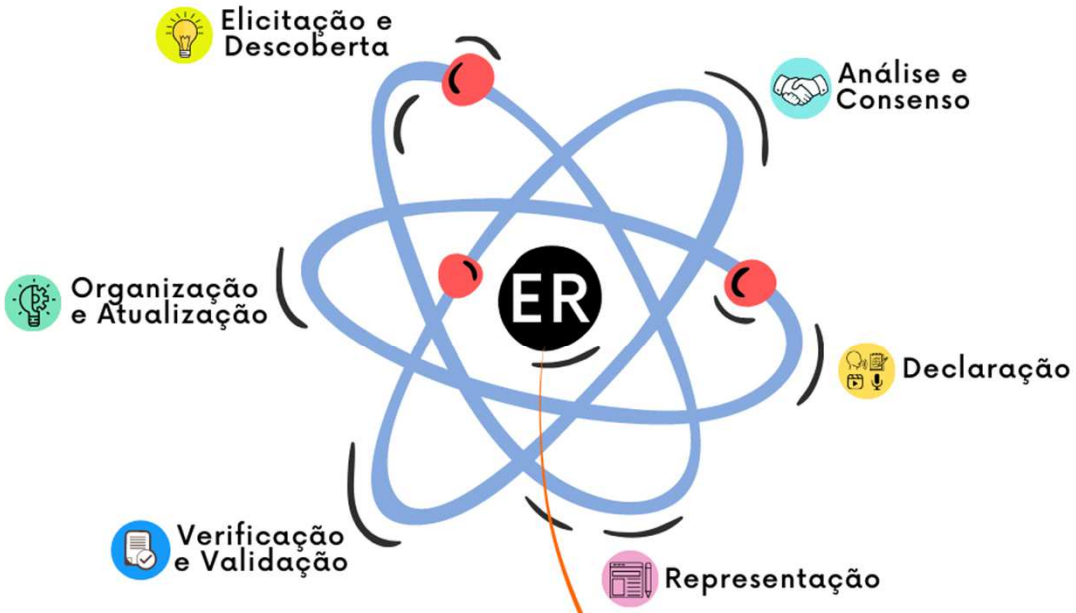
- É a *pedra fundamental* das relações que promovemos e sustentamos.
- A confiança, por vezes, é vista como algo que *simplesmente acontece* (mágico). Quando alguém olha para uma pessoa e diz: "*Essa pessoa parece ser de confiança*".
- Precisamos aprender a: vê-la, falar dela, fundamentá-la, construir e reconstrui-la.
- Não aprender, a partir da confiança pode fazer com que as relações se quebrem (ressentimento, descrença e insatisfação).

Confiança (cont.)

- Em relações maduras a confiança deve se tornar algo mais "*palpável*" e menos mágico, como, "*Eu sou capaz de dizer objetivamente, por que confio ou não em uma pessoa*".
- Ao olharmos para a confiança sobre um prisma mais objetivo, podemos fazê-lo diante de quatro aspectos:
  - Domínio
  - Sinceridade
  - Responsabilidade
  - Competência

Confiança (cont.)

Aspectos da Confiança	Exemplo
Domínio: é o contexto dentro do qual a confiança está sendo vista.	No trabalho, não confio no João.
Sinceridade: quando as pessoas estão falando a verdade ao comprometer-se.	Vejo que ele está falando a verdade quando se compromete com o planejamento.
Responsabilidade: quando as pessoas são responsáveis para fazer o que precisa ser feito (gerar ações)	Ele busca realizar as ações necessárias.
Competência: quando as pessoas possuem as competências necessárias para fazer o que precisa ser feito (gerar resultados).	Mas, ainda não possui as competências adequadas para realizar um planejamento efetivo.



Princípios

1. Orientação a valor
2. Comunicação dos requisitos
3. Satisfazer as partes interessadas
4. Entendimento compartilhado
5. Contexto da solução
6. Problema, requisitos e solução
7. Validação de requisitos
8. Mudança nos requisitos
9. Inovação da ER
10. Processo de ER
11. Impacto da ER

UnB

CEDIS Centro de Estudos, Desenvolvimento e Inovação em Software
cedis.unb.br

Prof. Dr. George Marsicano

Os 11 Princípios da ER

- **(P1) Orientação a valor**. Requisitos são meios para um fim, não um fim em si mesmo.
- **(P2) Comunicação dos requisitos**. ER não é sobre documentação e, sim, sobre comunicação. Os requisitos precisam ser comunicados adequadamente para conhecimento de todos os envolvidos (clientes, usuários, equipe técnica).
- **(P3) Satisfazer as partes interessadas**. ER é sobre satisfazer as necessidades das partes interessadas em relação a solução de software.
- **(P4) Entendimento compartilhado**. O desenvolvimento bem-sucedido de soluções é impossível sem um entendimento comum.
- **(P5) Contexto da Solução**. Os requisitos não surgem por acaso, sem contexto.

Os 11 Princípios da ER (cont.)

- **(P6) Problema, requisitos e solução**. Problema, requisitos e solução devem estar sempre entrelaçados.
- **(P7) Validação dos requisitos**. Requisitos não validados são inúteis.
- **(P8) Mudança dos requisitos**. A mudança nos requisitos, quando necessária, deve ser tratada de maneira simples e natural.
- **(P9) Inovação da ER**. Mais do mesmo não é suficiente.
- **(P10) Processo de ER**. Não podemos prescindir de um processo de ER inovador e adequado.
- **(P11) Impacto da ER**. A ER influencia as pessoas e o ambiente.

(P1) Orientação a valor

Requisitos são meios para um fim, não um fim em si mesmo

- O ato de **declarar** e/ou **representar** os requisitos não é um objetivo em si. Para serem úteis, os requisitos precisam agregar valor.

- Os *benefícios dos requisitos* referem-se ao quanto eles contribuem para:
  - A *redução* de *riscos* e *retrabalho* durante o desenvolvimento da solução; e
  - A construção de *soluções de sucesso* (satisfazem as necessidades dos stakeholders).

  - Os benefícios dos requisitos estão diretamente relacionados a sua *QUALIDADE* (necessário, singular, correto, sem ambiguidade, etc.). Ou seja, quanto *maior a qualidade do requisito menores os riscos e retrabalho* associados a ele, assim como *maiores as chances de construção de soluções de sucesso*.

(P1) Orientação a valor (cont.)

Requisitos são meios para um fim, não um fim em si mesmo

- O *valor da ER* pode ser considerado *cumulativo* (ao longo do tempo) e, principalmente, *indireto*, uma vez que estabelece o que a solução deve fazer, mas não são a solução em si.

(P2) Comunicação dos Requisitos

ER não é sobre documentação e, sim, sobre comunicação

- Os requisitos precisam ser *comunicados* adequadamente para *conhecimento de todos os envolvidos* (clientes, usuários, equipe técnica).
- Para realizar uma comunicação adequada a equipe de software precisa estar atenta as características dos *clientes*, do *produto*, da própria *equipe* e do *contexto de negócio*.
- A comunicação pode ser:
  - Quanto ao *meio*: face-a-face, áudio, vídeo, escrita, desenho;
  - Quanto a *temporalidade*: diária, a cada três dias, semanal, etc.;
  - Quanto a *forma*: formal e informal.

(P3) Satisfazer as partes interessadas

ER é sobre satisfazer as necessidades das partes interessadas

- O principal objetivo da ER é *entender* os desejos e necessidades das partes interessadas e *minimizar* o risco de entregar uma solução que não os atenda.
- As *partes interessadas* e seus papéis devem ser considerados para atingir esse objetivo.
- Cada parte interessada tem um papel no contexto da solução (por exemplo, usuário, cliente, operador, regulador, etc.) e, portanto, *diferentes pontos de vista sobre os requisitos*.

(P3) Satisfazer as partes interessadas (cont.)

ER é sobre satisfazer as necessidades das partes interessadas

- As partes interessadas devem ser classificadas quanto ao *grau de sua influência* (crítica, maior, menor) para o sucesso da solução.
- Isso ajuda na avaliação da *criticidade dos requisitos* e na gestão de *conflitos negociais*.
- É *vital* para o sucesso da ER:
  - *Envolver as pessoas certas* com papéis relevantes para a construção da solução;
  - *Identificar inconsistências e conflitos entre os requisitos de diferentes partes interessadas* e resolvê-los, seja encontrando um consenso, anulando ou especificando variantes da solução as para partes interessadas com necessidades diferentes.

(P4) Entendimento compartilhado

Um entendimento comum é a base para a construção de soluções

- Um processo de ER, geralmente, envolve múltiplas pessoas (stakeholders, engenheiros de software, etc.) e fontes de informação.

- É **responsabilidade da ER** buscar estabelecer um *entendimento compartilhado* entre essas pessoas quanto ao *problema / oportunidade*, *necessidades* e *requisitos*.

- O entendimento compartilhado pode ser:
  - *Explícito*: frequentemente utilizado em processos orientados a planos, onde, por exemplo, os requisitos são mais *detalhadamente declarados e escritos*.
  - *Implícito*: muitas vezes utilizado em processos ágeis, nos quais, por exemplo, os requisitos são amplamente *declarados em conversas face-a-face*, sem serem escritos em detalhes.

UnB

CEDIS Centro de Estudos, Desenvolvimento e Inovação em Software
cedis.unb.br

(P4) Entendimento compartilhado (cont.)

Um entendimento comum é a base para a construção de soluções

- Visando estabelecer um entendimento compartilhado pode-se usar práticas, como:
  - Construção de *glossários* e *protótipos*
  - Utilização de *sistemas existentes* como referência
  - Fornecer exemplos de *resultados esperados*
  - Realizar *ciclos curtos de feedback*.

(P4) Entendimento compartilhado (cont.)

Um entendimento comum é a base para a construção de soluções

- Alguns fatores podem ser vistos como habilitadores ou obstáculos de um entendimento compartilhado, por exemplo:

Habilitadores	Obstáculos
Conhecimento do domínio	Distância geográfica
Compartilhamento de cultura e valores	Restrições regulatórias
Domínio de padrões específicos	Terceirização
Confiança mútua	Falta de confiança mútua
Existência de soluções de referência	Alta rotatividade de pessoas envolvidas

(P5) Contexto da solução

Os requisitos não surgem por acaso, sem contexto

- *As soluções de software estão sempre inseridas em um contexto maior*. Sem entender esse, é quase *impossível* estabelecer, corretamente, os *requisitos* de uma solução.

- O *contexto* de uma solução é definido como a parte do ambiente de um sistema (*domínio da aplicação*) que é relevante para a compreensão da solução e seus requisitos.
  - Exemplo: Um login de usuário para um sistema bancário pode apresentar requisitos diferentes de um login para a área de membros de um clube esportivo.

(P5) Contexto da solução (cont.)

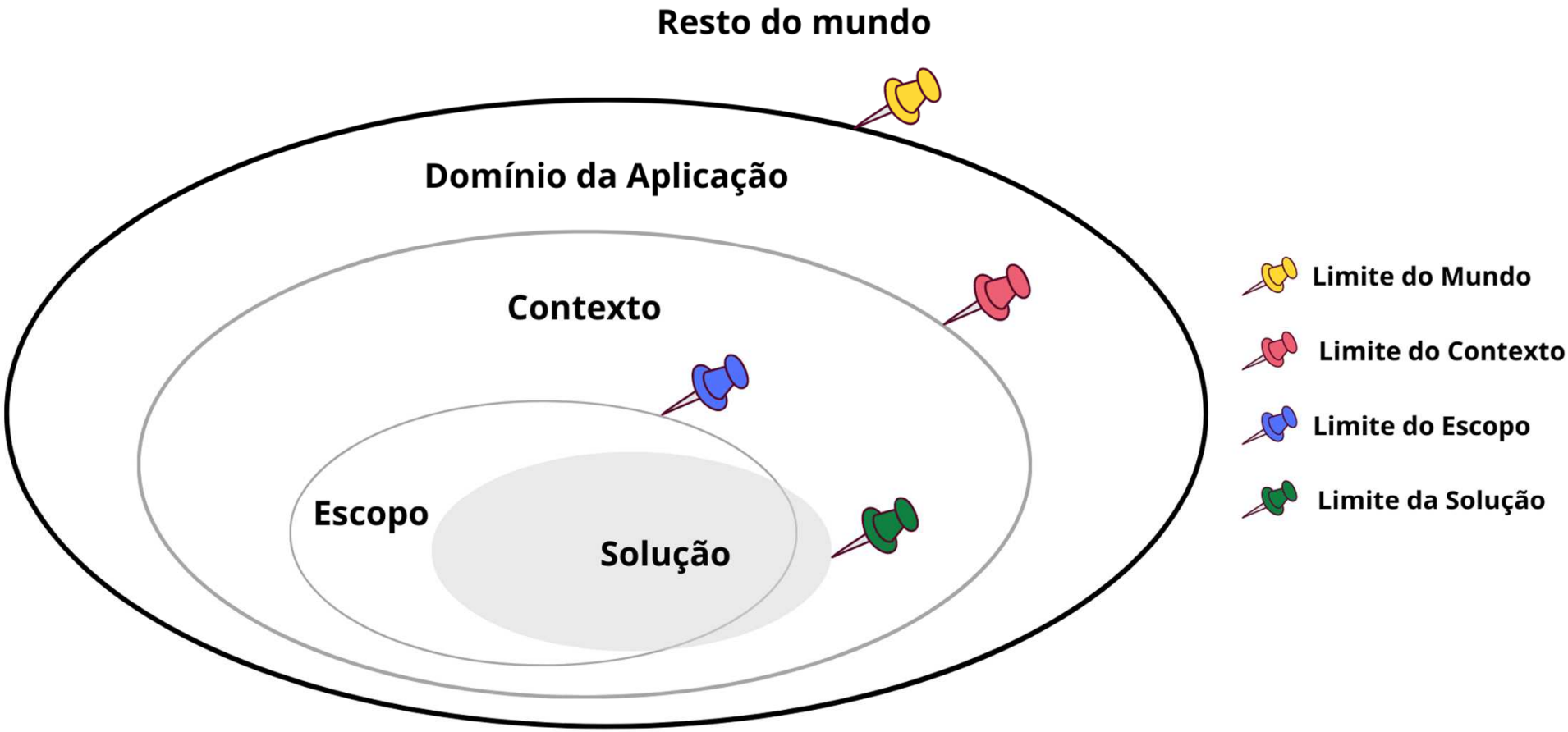
Os requisitos não surgem por acaso, sem contexto

- Compreender o contexto da solução:
  - É importante para *identificar as interfaces* relevantes para sistemas existentes, *regulamentos* e *partes interessadas* da solução pretendida.
  - Ajuda a fixar o *limite do contexto* (definir qual parte do contexto da solução deve ser considerada ou não).
  - Ajuda também fixar o *limite da solução* (definindo o que está dentro do escopo da solução pretendida e o que não está).

- É importante entender que a ER vai além de considerar requisitos no limite da solução. *ER também tem que lidar com aspectos que vêm do contexto do sistema.*

(P5) Contexto da solução (cont.)

Os requisitos não surgem por acaso, sem contexto



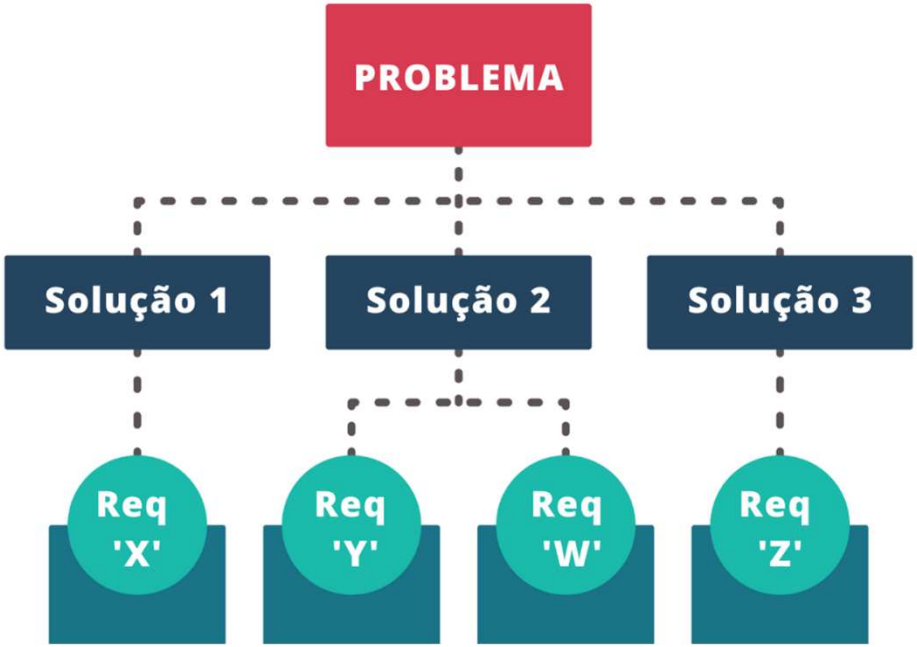
UnB

CEDIS Centro de Estudos, Desenvolvimento e Inovação em Software
cedis.unb.br

(P6) Problema, requisitos e solução

Problema, requisitos e solução devem estar sempre entrelaçados

- *Problemas, suas soluções e requisitos estão intimamente interligados*. Qualquer situação em que as pessoas estejam insatisfeitas com processos e resultados pode ser considerada um problema.



UnB

CEDIS Centro de Estudos, Desenvolvimento e Inovação em Software
cedis.unb.br

(P6) Problema, requisitos e solução (cont.)

Problema, requisitos e solução devem estar sempre entrelaçados

- Para eliminar o problema, um sistema sociotécnico pode entrar em ação. Este sistema precisa de *requisitos* para tornar o sistema uma *solução* para o *problema*.
- O projeto de um sistema inovador pode ser conduzido por uma solução que acaba resolvendo um *problema* conhecido.
- A *compreensão do problema e da solução*, por sua vez, leva à *elaboração de requisitos que satisfaçam as necessidades do usuário* e sejam implementados em uma *solução concreta*.

(P7) Validação de requisitos

Requisitos não validados são inúteis

- A validação é uma *atividade central da ER* e descreve o processo de *confirmação* de que os *requisitos* declarados *são corretos* e correspondem às necessidades das partes interessadas.

- Para a validação, é crucial que:
  - Os conflitos entre as partes interessadas tenham sido resolvidos e as prioridades definidas (*consenso*);
  - As necessidades das partes interessadas estejam cobertas pelos requisitos.

(P8) Mudança nos requisitos

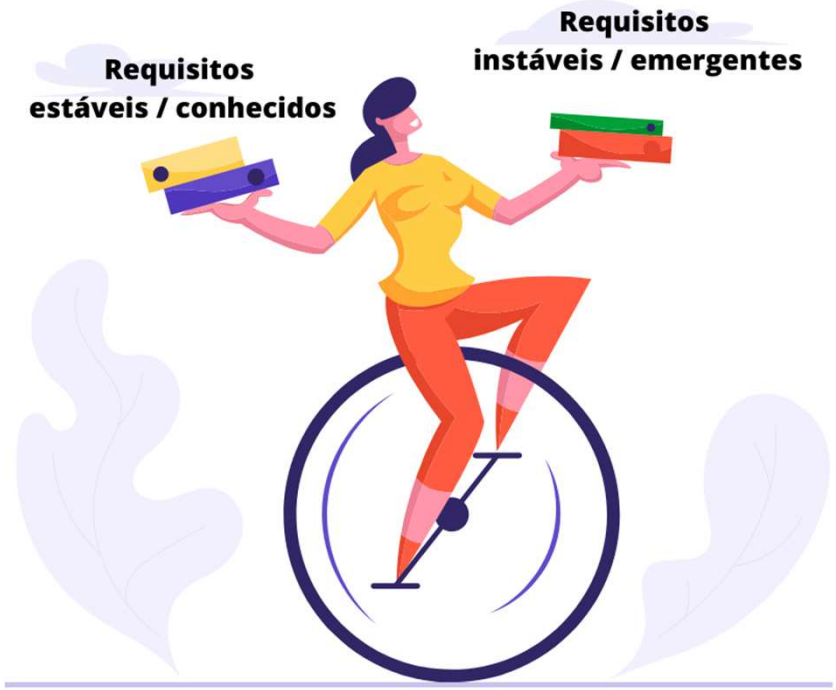
A mudança nos requisitos deve ser simples e natural.

- Necessidades, negócios e capacidades estão sempre expostos à evolução. Como consequência, os requisitos para soluções que satisfaçam necessidades, suportem negócios e usem recursos técnicos também mudarão.
- *Se os requisitos não acompanham a evolução do sistema, eles eventualmente perderão valor e se tornarão inúteis*.
- As *razões para as mudanças* são vastas e podem variar, por exemplo:
  - Processos de negócios alterados
  - Detecção de erros ou domínio incorreto de algumas premissas
  - Mudança de tecnologia
  - Novos produtos concorrentes no mercado

(P8) Mudança nos requisitos (cont.)

A mudança nos requisitos deve ser simples e natural.

- Por um lado, os engenheiros de software devem *permitir que os requisitos mudem*, pois ignorar sua necessidade de mudança seria um erro.
- Por outro lado, eles precisam *manter os requisitos minimamente estáveis*, pois o custo da mudança pode se tornar alto e o desenvolvimento do produto pode não ocorrer de maneira sistematicamente com mudanças constantes nos requisitos.
- *É preciso buscar equilibrar, mudança e estabilidade.*



UnB

CEDIS Centro de Estudos, Desenvolvimento e Inovação em Software
cedis.unb.br

(P9) Inovação da ER

Mais do mesmo não é suficiente

- A principal preocupação do ER é satisfazer as necessidades das partes interessadas. No entanto, é crucial *não cair no papel de gravador de voz do stakeholder*, declarando e representando exatamente o que os stakeholders dizem, pois isso significaria *perder a oportunidade* de fazer *coisas novas*, *melhores do que antes*, além de *abrir da responsabilidade pela solução.*

- Os engenheiros de software / requisitos devem caminhar na linha tênue entre *permanecer inovadores* sem acreditar que sabem tudo melhor do que as partes interessadas.

- Em pequena escala, a ER *molda sistemas inovadores*, buscando *novos recursos empolgantes* e *facilidade de uso*.

UnB

CEDIS Centro de Estudos, Desenvolvimento e Inovação em Software
cedis.unb.br

(P9) Inovação da ER (cont.)

Mais do mesmo não é suficiente

- Os engenheiros de software / requisitos também precisam olhar para o quadro geral *explorando*, com as partes interessadas, se existem *maneiras disruptivas de fazer as coisas*, levando à inovação em larga escala.

(P10) Processo de ER

O processo de ER de ser inovador e adequado

- A ER pode ser considerada *a mais humana das disciplinas* da Engenharia de Software. Nela, competências técnicas e humanas estão fortemente envolvidas.

- Seu processo de trabalho *não* deve ser visto *apenas tecnicamente*, mas *também* como uma abordagem de *como lidar com pessoas* de diferentes características e perspectivas.

  - *Tecnicamente*, seu processo deve ser sistemático e disciplina;
  - *Humanamente*, sua abordagem com as pessoas deve ser flexível e adaptável.

(P10) Processo de ER (cont.)

O processo de ER de ser inovador e adequado

- Assim como na Engenharia de Software, **não há um "modelo único" para um processo de ER**, aplicável a todos os contextos.
  - Os engenheiros de software / requisitos:
    - Precisam conceituar os processos de ER adequados para o problema em questão.
    - Não devem sempre usar o mesmo processo, práticas e produtos de trabalho, mas selecionar aqueles que melhor ajudam com o problema, contexto e ambiente de trabalho em questão.
- Lembre-se:
  - *Seu processo pode ser tão iterativo quanto quiser, mas ainda precisa entender o que o negócio necessita.*

(P11) Impacto da ER

A ER influencia as pessoas e o ambiente

- O processo de *ER auxilia a promover*, entre os envolvidos, uma *reflexão* e *amadurecimento* sobre problema, necessidades e possíveis impactos que a solução de software poderá gerar em seu ambiente.
  - *Compreensão do Problema*: A ER, poderá mudar a maneira como os envolvidos pensam seu problema, agora ou depois.
  - *Impacto no Ambiente*: A ER, estabelece o que o software irá fazer, suas restrições e potencialidades, as quais irão impactar no ambiente social no qual estará inserido.

Requisitos de Software

Valores e Princípios da Engenharia de Requisitos

georgemarsicano@unb.br

George Marsicano

UnB

CEDIS Centro de Estudos, Desenvolvimento e Inovação em Software