

NÁSKOK
DÍKY
ZNALOSTEM

PROFINIT

Requirements Engineering

Sběr a analýza požadavků

Kolektiv autorů

Říjen 2018

Requirements Engineering

1. Úvod

- Proč Requirements Engineering?
- Requirements / požadavky

2. Requirements Engineering

- **Proces**
- Plánování
- Změny požadavků

3. Postřehy z praxe

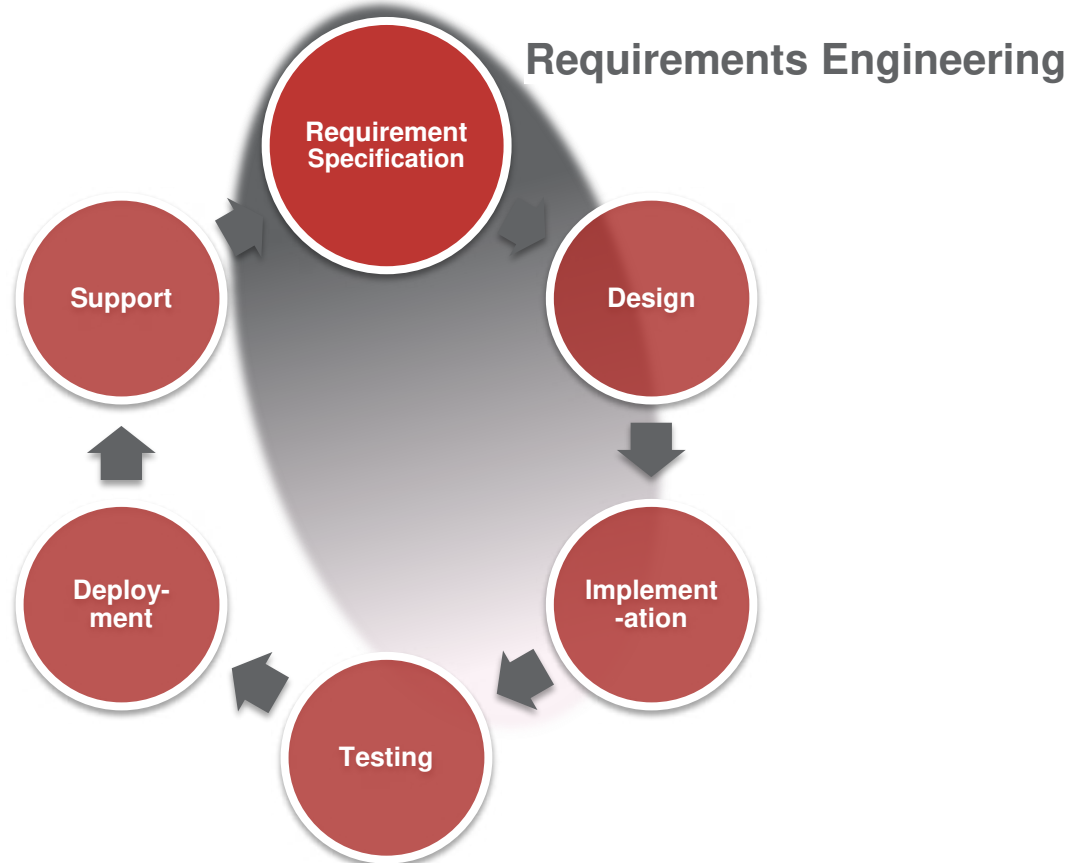
› **CÍL: Praktické principy “zlaté střední cesty”.**

- Použití v rozsáhlejších i menším kontextu
- Žádná informace není absolutní pravdou

Requirements Engineering

Requirements Engineering is a systems and software engineering process which covers all of the activities involved in discovering, documenting and maintaining a set of requirements for a computer-based system.

Kotonya G. and Sommerville, I. Requirements Engineering: Processes and Techniques. Chichester, UK: John Wiley & Sons. 1998



Requirements Engineering - proč?

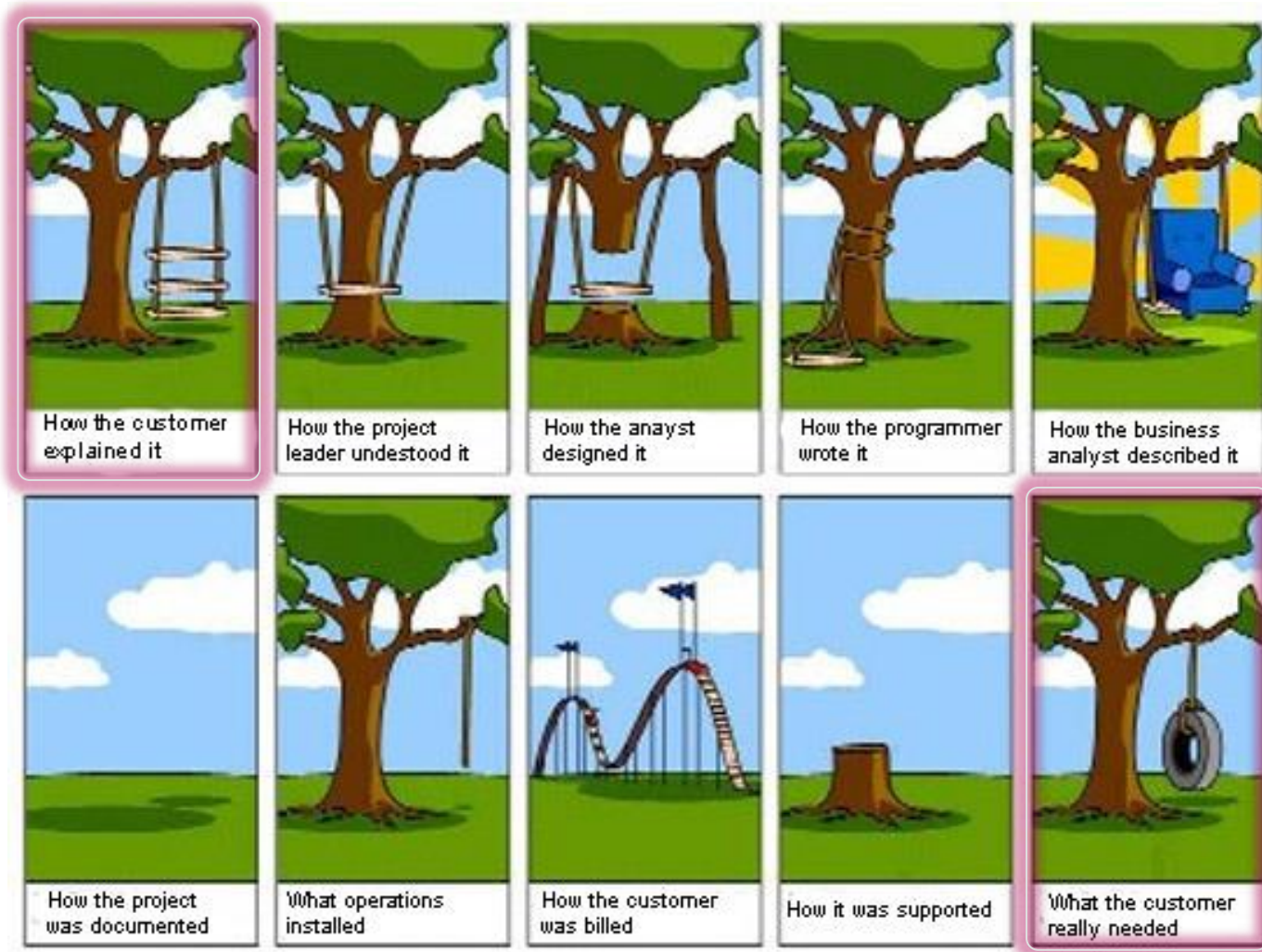
- › Specifikace požadavků
 - Základní část dokumentace systému
 - Zadání pro design a pro vývoj
 - Definice rozsahu dodávky ... kontrakt

- › Rozsah je parametrem ceny díla
 - Neuhlídaný rozsah = nízká profitabilita projektu (ziskovost/ztrátovost)

- › Špatně definované požadavky způsobují **neúspěch projektů**
 - http://www.it-cortex.com/Stat_Failure_Rate.htm
 - <http://www.zdnet.com/blog/projectfailures/cio-analysis-why-37-percent-of-projects-fail/12565>

Requirements Engineering - proč?

- › Špatně definované požadavky způsobují **neúspěch projektů**



Requirements / požadavky

Requirements Engineering is a systems and software engineering process which covers all of the activities involved in discovering, documenting and maintaining a set of requirements for a computer-based system.

Kotonya G. and Sommerville, I. Requirements Engineering: Processes and Techniques. Chichester, UK: John Wiley & Sons. 1998

Obecně:

Requirements are written statements that specify

- **capabilities** needed to solve a problem or to achieve an objective
- **conditions** of a delivered system, service, product, or process
- **constraints** on the system, service, product, or process

*A Guide to the Project Management Body of Knowledge (PMBOK), www.pmi.org
Institute of Electrical and Electronic Engineers (IEEE), www.ieee.org*

IT:

Requirements are a specification of what should be implemented.

They are descriptions of **how the system should behave**, or of a system **property** or **attribute**. They may be a **constraint** on the development process of the system.

Ian Sommerwille, Peter Sawyer, Cutter IT Journal, May 2000

Requirements / požadavky

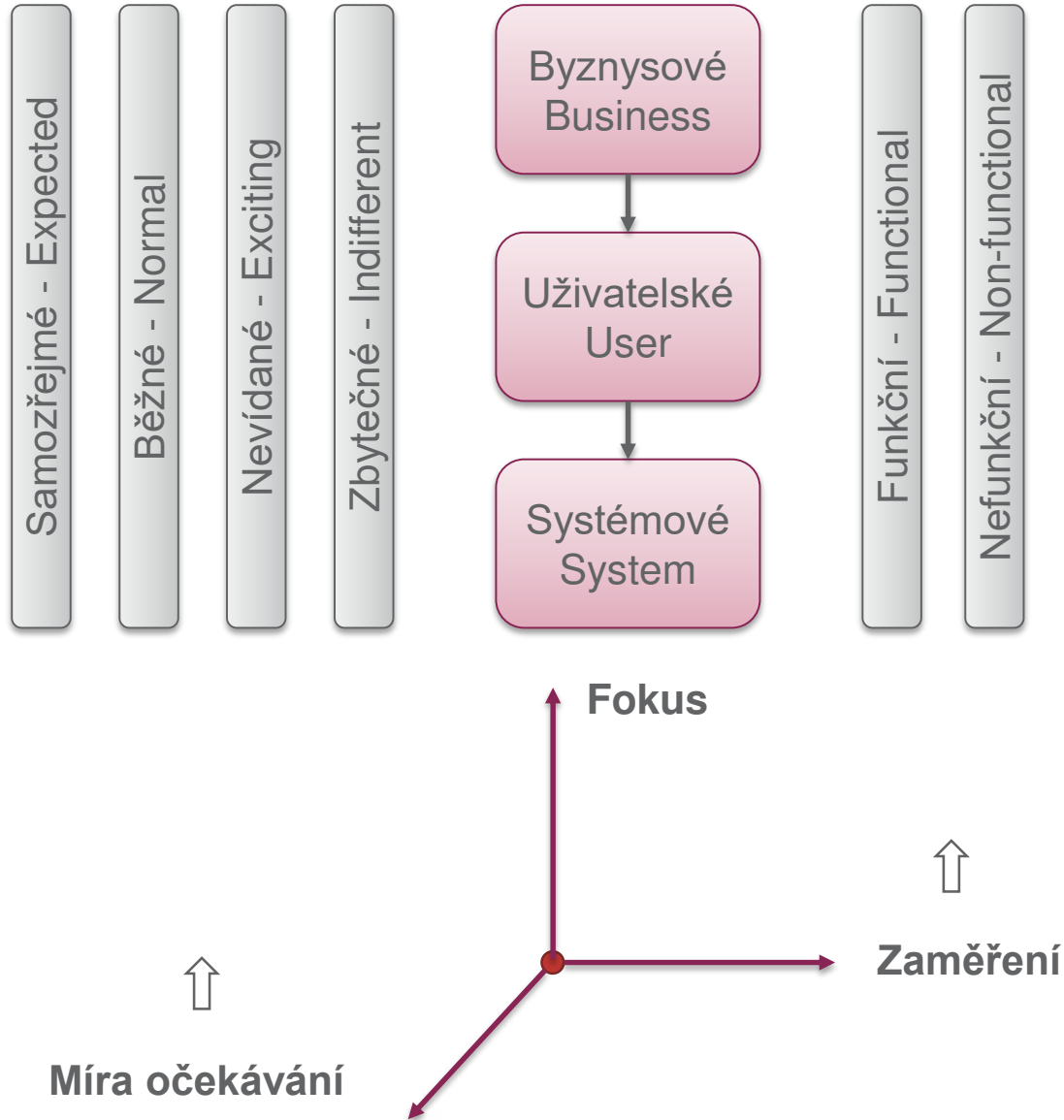
- › Věcný obsah požadavku
 - **Aktor** („capability“) – osoba, událost, věc, která provádí akci
 - uživatel, zákazník, systém
 - **Akce** („capability“) – sloveso, které popisuje, co aktor provádí
 - zobrazí, označí, stiskne
 - **Vymežující kritéria** („conditions & constraints“) – kvantitativní nebo kvalitativní
 - do 2 sekund, pomocí myši, v novém okně

- › Například
 - Uživatel otevře obrazovku “Vyhledání odběratele”
 - Uživatel musí zadat IČO nebo část názvu a myší stiskne tlačítko “Hledat”
 - Systém musí do 3 sekund zobrazit seznam odběratelů, kteří vyhovují zadanému kritériu

Nebo také...

 - Povinná pole musí být viditelně označena
 - Front-end systému musí mít responzivní design
 - Systém musí mít kontextovou nápovědu v českém jazyce na úrovni polí

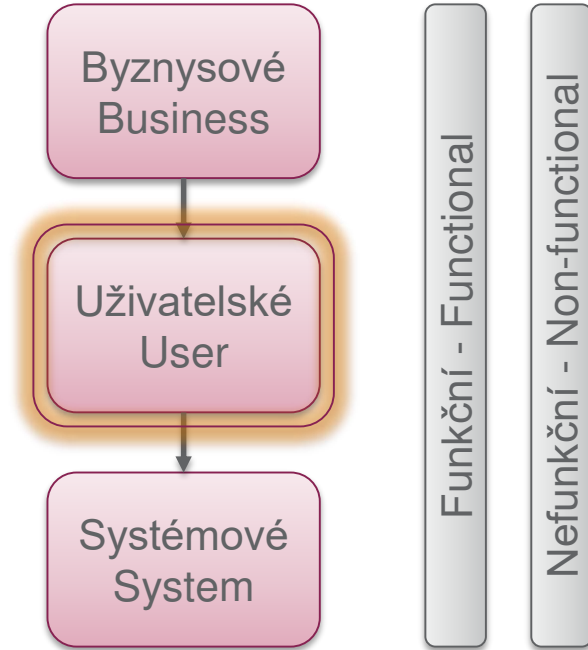
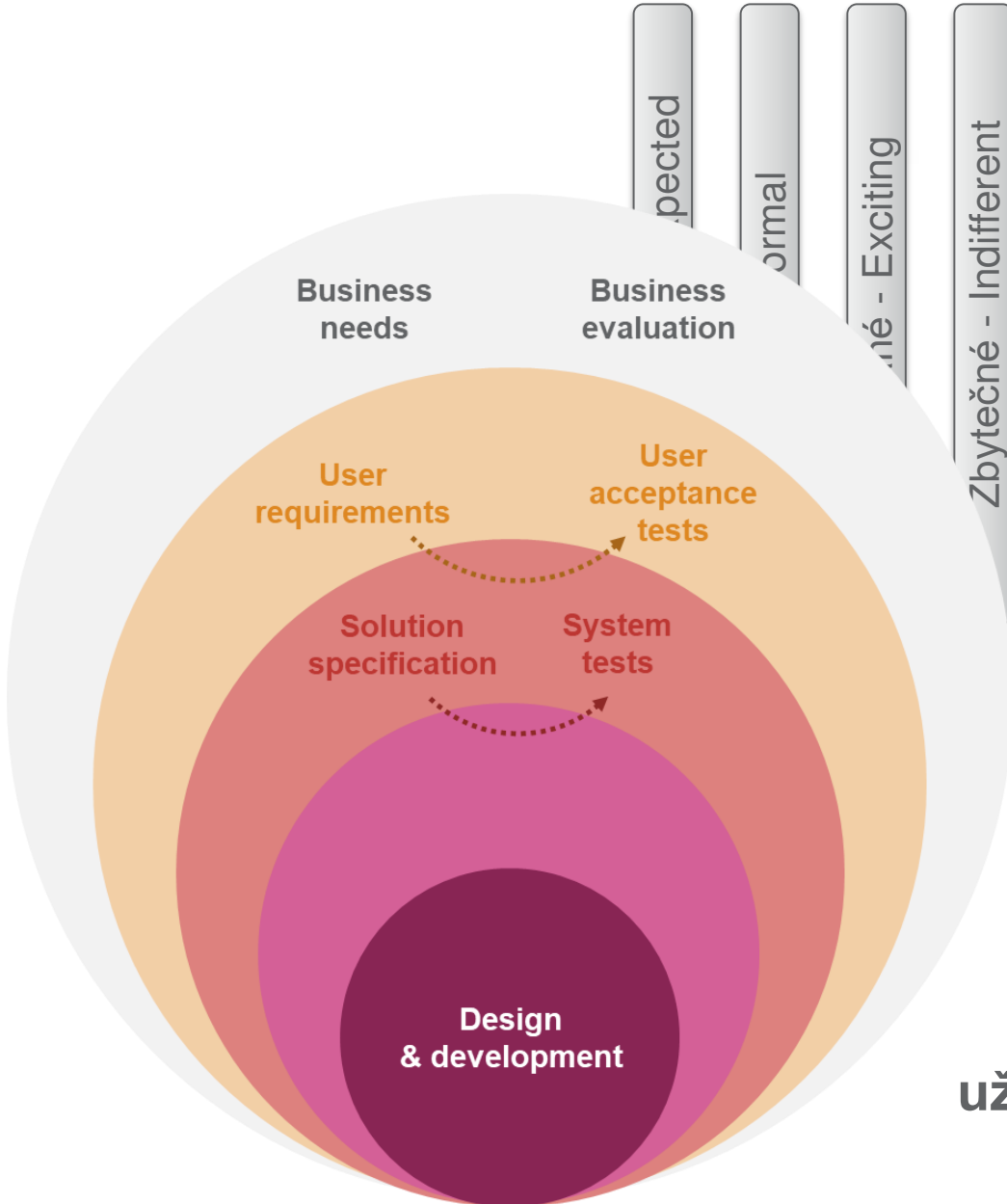
Typy požadavků



Typy požadavků – příklady

- › Pobočka sama rozhodne o komunikačním kanálu s klientem
 - Byznysový / funkční / běžný
- › Systém musí oddělovat obchodní činnosti od pracovních pozic
 - Byznysový / funkční / běžný? nevídaný? samozřejmý?
- › Systém musí být dispoziční v režimu 24x7
 - Byznysový / nefunkční / běžný? nevídaný?
- › Uživatel otevře obrazovku “Vyhledání odběratele”
 - Uživatelský / funkční / běžný
- › Uživatel musí zadat IČO nebo část názvu a myší stiskne tlačítko “Hledat”
 - Uživatelský / funkční / běžný
- › Front-end systému musí mít responzivní design
 - Uživatelský / nefunkční / běžný? nevídaný? samozřejmý?
- › Povinná pole musí být viditelně označena
 - Uživatelský / nefunkční / běžný? nevídaný?

Typy požadavků vs. SDLC



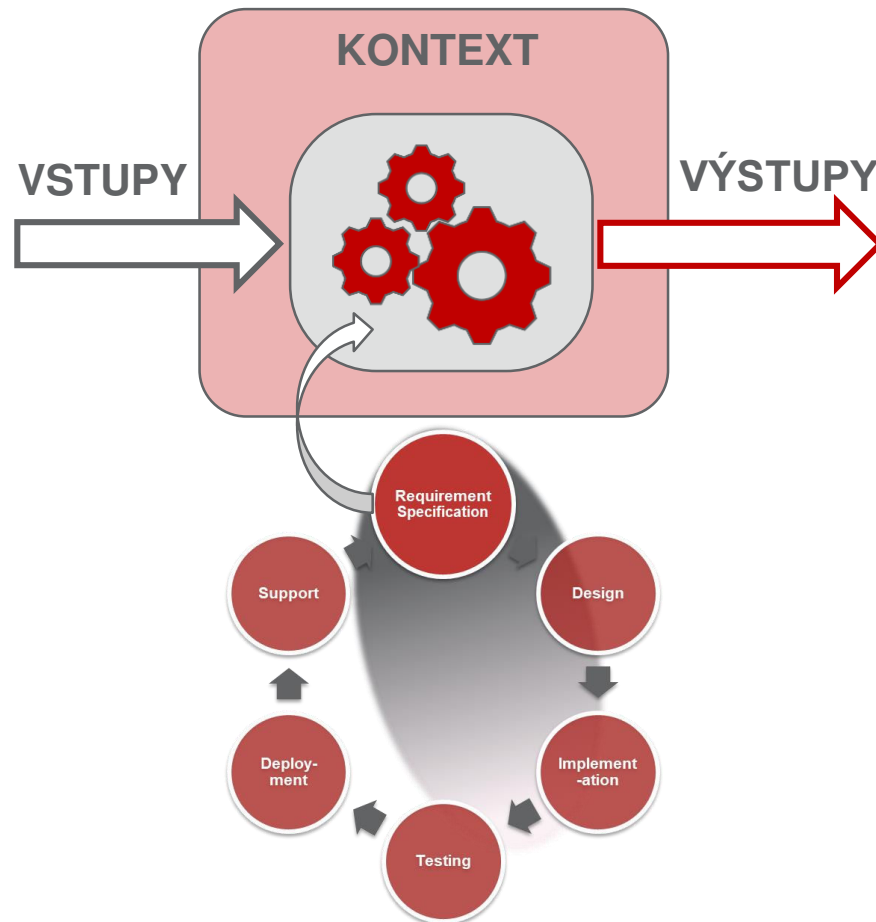
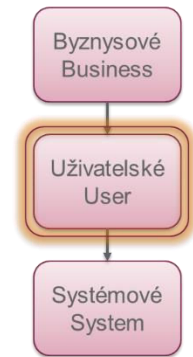
Dále se budeme zabývat
uživatelskými požadavky

Requirements Engineering

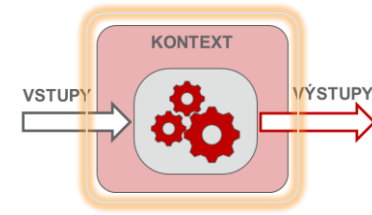
- proces, plánování, změny -

Requirements Engineering

- › *Requirements Engineering* is a systems and software engineering process which covers ...
Posloupnost aktivit, která se odehrává v daném **kontextu** a transformuje množinu **vstupů** na množinu **výstupů**



Requirements Engineering – kontext



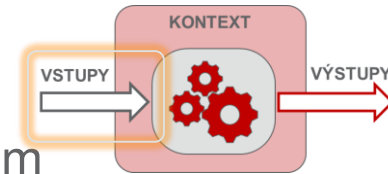
› Kontext = “big picture”

- Co je očekáváno, za jakých podmínek, jaká jsou omezení ... 5 W's
 - **What & Why** – co má vzniknout a proč, co je pro zadavatele kritériem úspěchu?
 - změna technologie?, snížení nákladů?, uvedení na trh do xxx?, ...
 - “Zrušit X% dosavadních aplikací a snížit IT náklady o Y Kč ročně”
 - “Vytvořit internetovou aplikaci, která klientům poskytne komfortní prostředí internetové samoobsluhy”
 - **Who** – kdo je zadavatelem, kdo je v tématu **zainteresován**
 - **Stakeholders** ... jakou mají roli, kdo bude akceptovat požadavky?
 - Zadavatel, sponzor, manažer s rozhodovací pravomocí, “šampión”, ...
 - **Where & When** – projektová aj. omezení, existující standardy, zvyklosti, ...
 - Termíny, nástroje, metodologie, ...

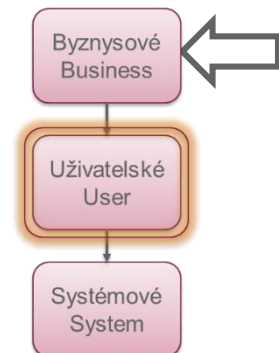
› Kde se to dozvědět?

- Typicky existuje nějaká forma zadávací dokumentace na úrovni business requirements → **vstupy**
- Zním situaci
- Zeptám se

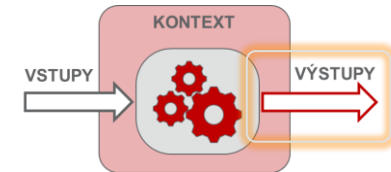
Requirements Engineering – vstupy



- › Pomineme-li kontext, pak asi nejdůležitějším vstupem do procesu specifikace uživatelských požadavků je **zadání a vymezení rozsahu** neboli **scope** ... čeho se specifikace uživatelských požadavků má týkat, a čeho nikoliv
- › **Obchodní požadavky** (business requirements) v “jazyce” zadavatele
- › Nepříliš formalizovaná zadání
 - Věta: “Změnit uživatelské role a rozsah jejich oprávnění”
 - Odstavec: “*Systém bude načítat ... platební transakce a automatizovaně je porovnávat s pohledávkami. Výsledkem budou seznamy spárovaných a nespárovaných transakcí. Výsledek bude uložen v systému s možností zobrazení uživatelem. Nad nespárovanými transakcemi může uživatel realizovat ...*”
- › Formalizovaná zadání
 - **Business Requirements Document** – BRQ / BRD, zvaný též
 - “Operational Concept Description” – OCD,
 - “Project Product Definition” – PPD, ...
 - Typicky obsahuje
 - Seznam nebo textový popis byznysových požadavků
 - Někdy i seznam high-level nefunkčních požadavků
 - Další informace, které se zpravidla týkají kontextu

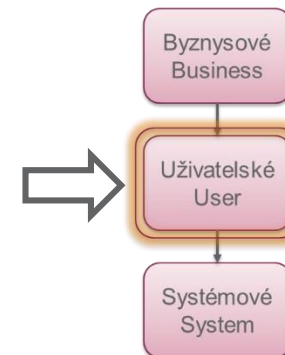
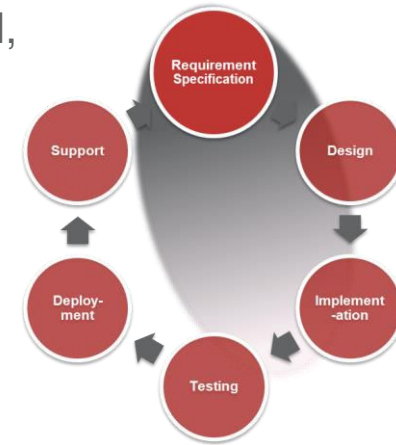


Requirements Engineering – výstupy

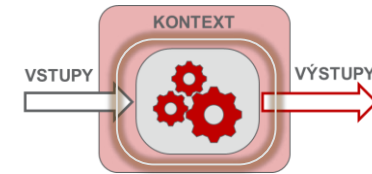


- › Výstupem je **specifikace uživatelských požadavků** ve formě jednoho nebo více dokumentů, případně dalších artefaktů
 - Za použití výrazových prostředků, kterým rozumí uživatel, a také další účastníci softwarového procesu

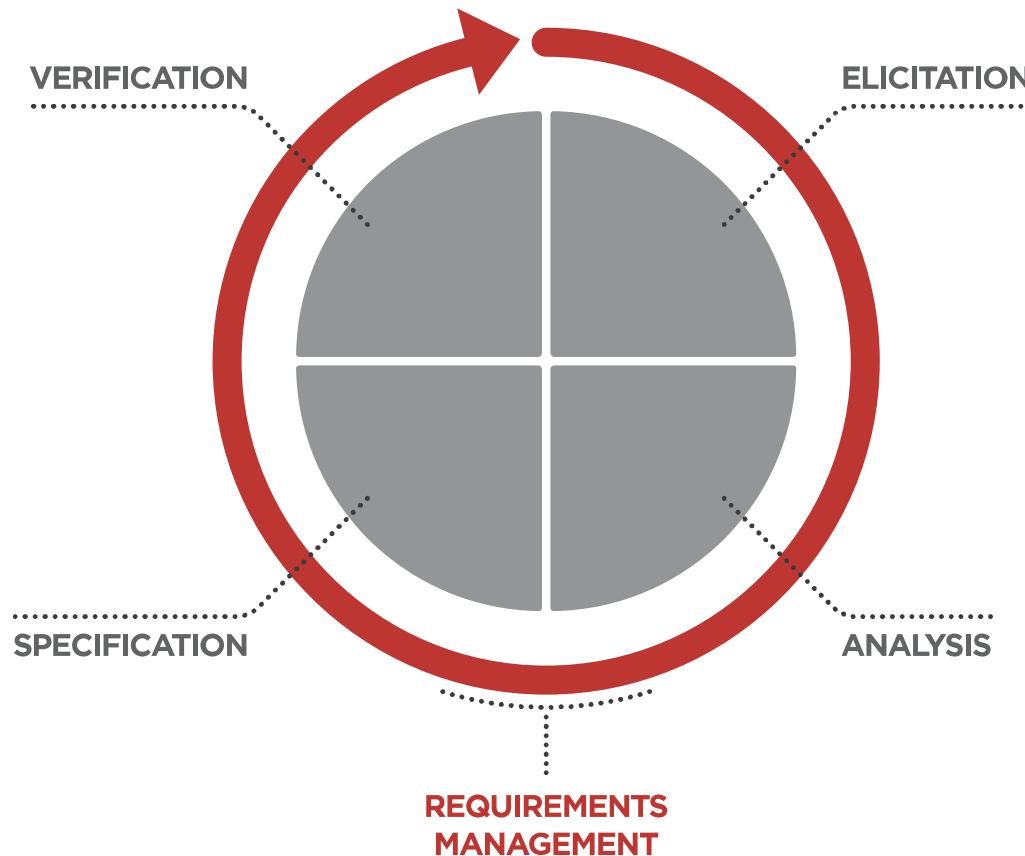
- › Formát dokumentace
 - Pouhý **katalog** funkčních požadavků
 - Spíše výjimečně
 - Zpravidla **funkční specifikace** ve formátu, který je typicky dán kontextem
 - Uživatelsky srozumitelná!
 - plus **katalog nefunkčních požadavků**
 - Mnohdy i **model** budoucího systému
 - Tzv. “mockup”



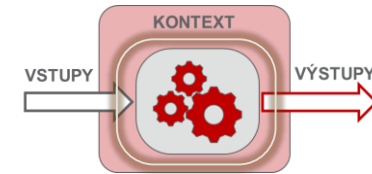
Requirements Engineering – proces



- › Cíl = definovaným postupem
 - **zjistit** uživatelské požadavky,
 - **systematicky** je **popsat** ve formátu, který je očekáván, a
 - **zvalidovat** je



Requirements Engineering – proces



› Proces = 4 fáze specifikace uživatelských požadavků

– Elicitation

- Vyzvídání, zjišťování, shromažďování informací
- Výstup = “stated requirements”

– Analysis

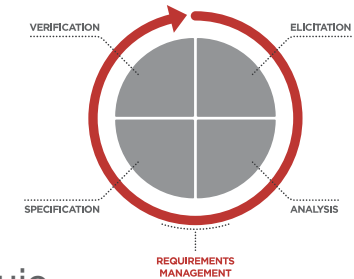
- Získání systematické představy o tom, co je skutečně potřeba
- Výstup = “real requirements”

– Specification

- Zdokumentování “real requirements” tak, jak je v daném kontextu vyžadováno
- Výstup = “documented requirements”

– Verification / Validation

- Ověření, zda specifikované požadavky jsou věcně a formálně správné
- Výstup = “verified/valid requirements”



› Plus management požadavků

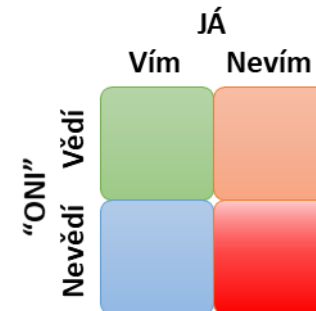
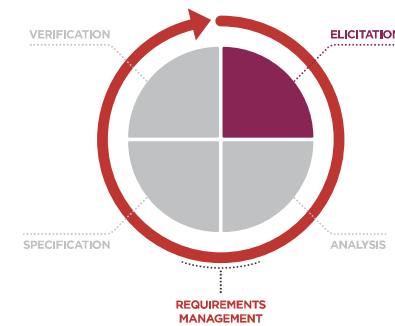
› Fáze procesu specifikace se mohou iterativně opakovat

- Zpravidla tomu tak je

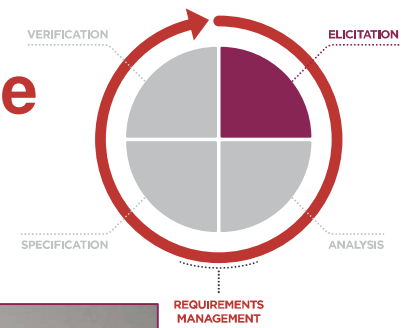
› Management požadavků je víceméně průběžná činnost

Requirements Elicitation

- › Vyzvídání, zjišťování, shromažďování informací
 - “Data gathering”
- › Zdroje informací
 - Stakeholders ← kontext ... typicky zástupci uživatelů
 - Existující podklady – dokumenty, vnitřní předpisy, internet
- › Existuje mnoho technik, důležité je:
 - Jak to funguje dnes?
 - Jak by to mělo fungovat? ... interview, workshop
 - Zrcadlo – “tomu, co říkáte, rozumím takto ...”
→ whiteboard, fixy, Post-It Notes
- › Výstup → nestrukturované informace - **stated requirements**
 - “How the customer explained it”
- › Výzva → “nevíme, že to nevíme” ... Johari Window
 - Často se to týká samozřejmých a nevídaných požadavků
 - Klást otevřené otázky → nutí k přemýšlení
 - Postupovat ve spirále – vracet se



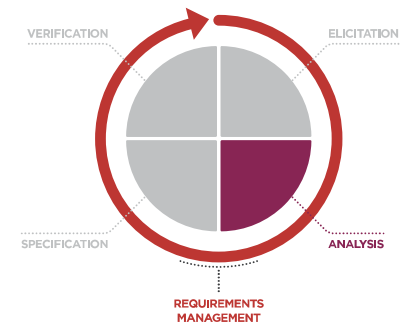
Requirements Elicitation – osvědčilo se



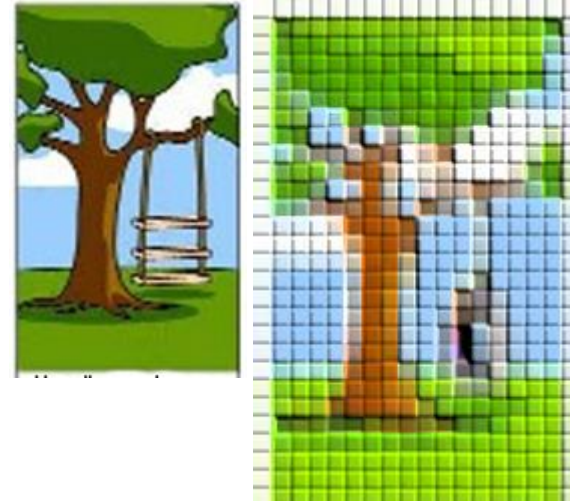
- › Whiteboard, sada fixů a Post-It Notes



Requirements Analysis

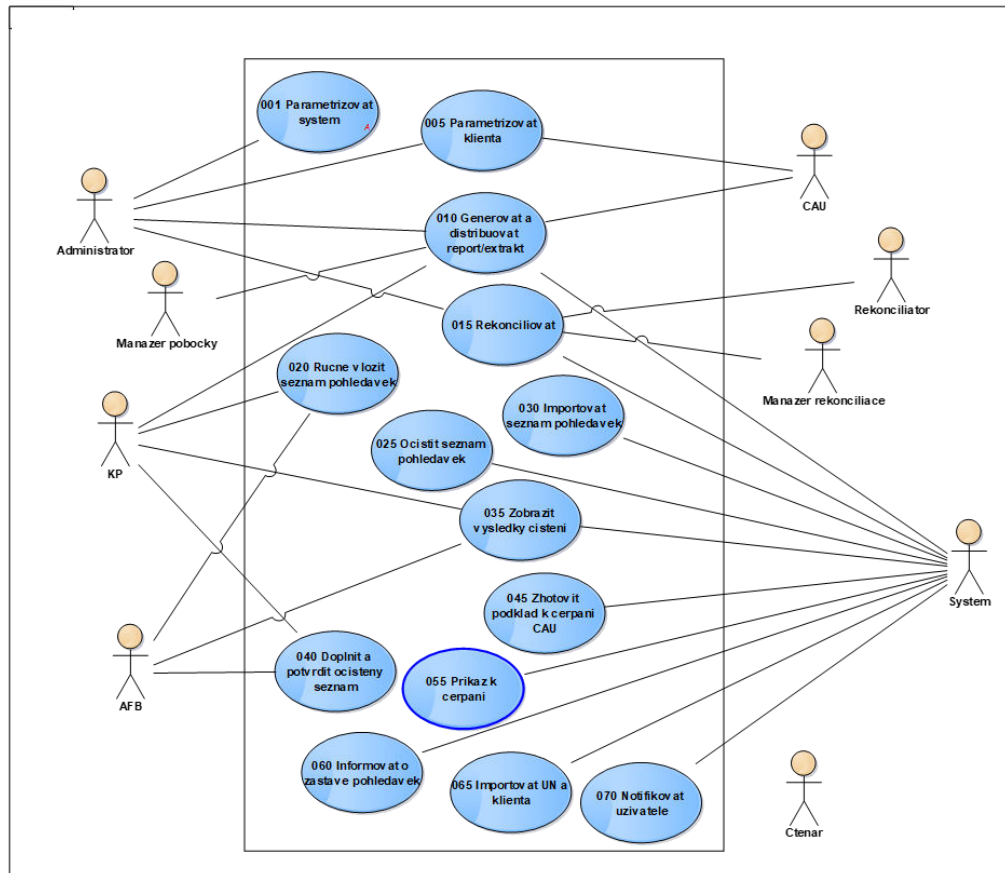


- › Zpracování “stated requirements” s cílem dát jim smysl
- › Vytvořit si systematickou představu o tom, co klient skutečně potřebuje
- › Existuje mnoho technik, osvědčilo se:
 - **Profilování uživatelů** (user profiling)
 - Skupiny (profily, role) budoucích uživatelů, co dělají a co pro to potřebují
 - Znázornění – případy užití (use cases)
 - **Modelování procesů** (process modelling)
 - Akce, které provádějí jednotlivé skupiny uživatelů, jejich pořadí, závislosti a logické vazby
 - Znázornění – procesní mapy (workflows) → ideálně “swim lane” diagramy
 - **Dekompozice požadavků** (requirements breakdown)
 - Jaká je logická struktura požavků
 - Znázornění – strom požadavků – mentální mapa nebo jiný diagram, příp. Excel
- › Výstup → strukturované informace - **real requirements**



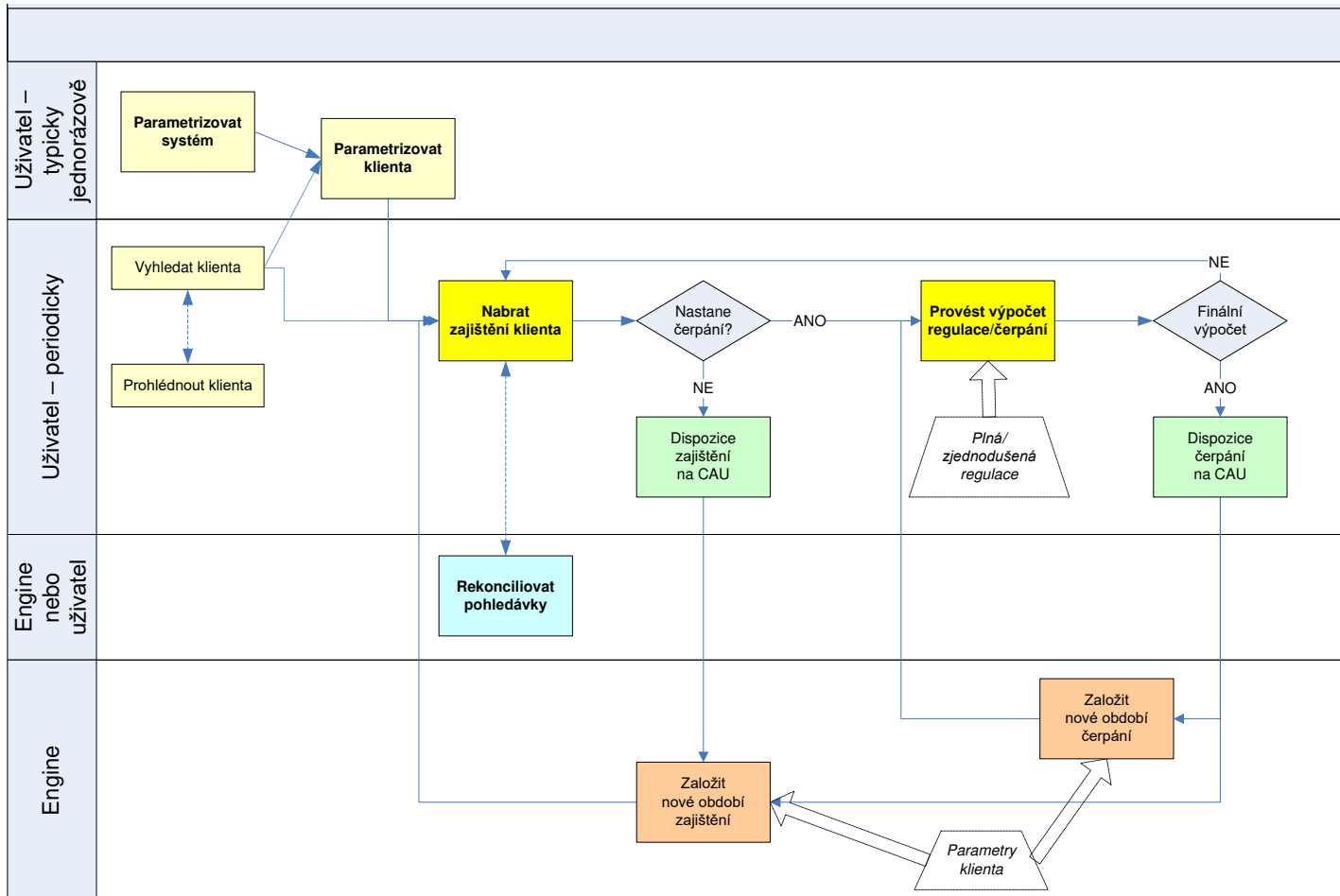
Requirements Analysis

› Profilování uživatelů – Use Cases



Requirements Analysis

- › Modelování procesů – “swim lane” diagram



Requirements Analysis

› Dekompozice požadavků

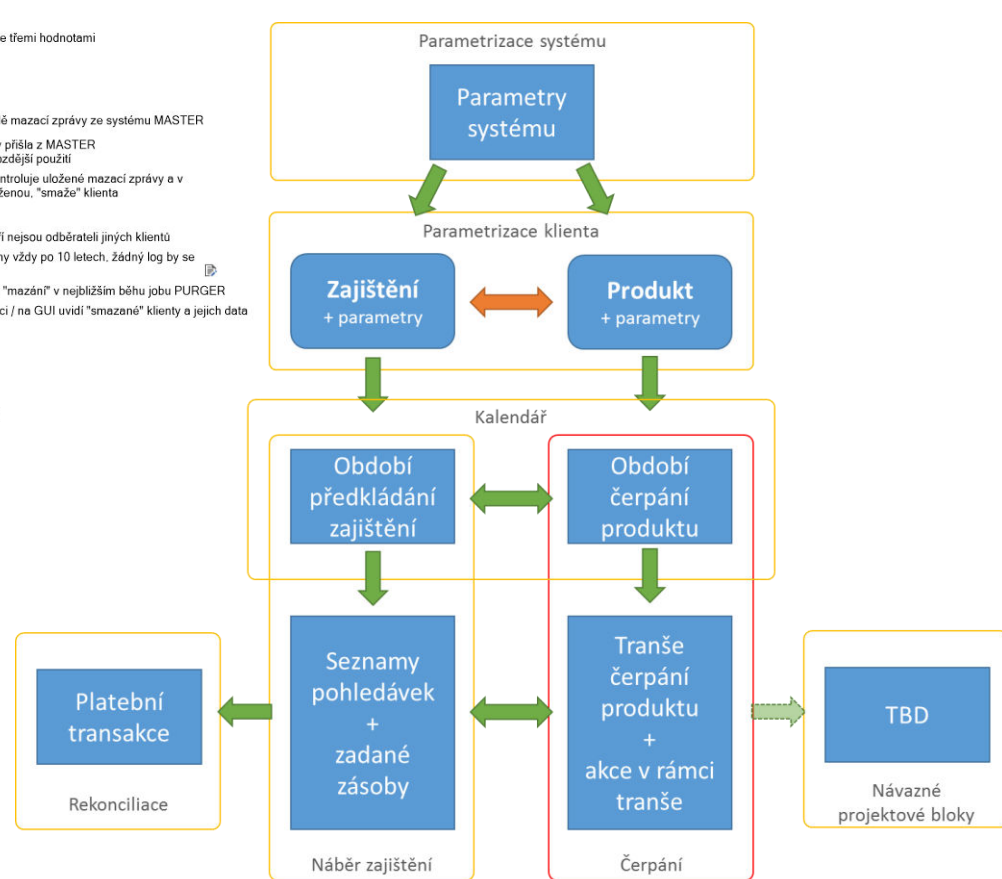
Požadavky

- Ukládá pouze minimální množinu osobních údajů.
- Ukládá pouze relevantní osoby.
- Udrží svoje osobní údaje aktuální.
- Systém používá osobní údaje oprávněně. ○ COPRYS: Omezit vyhledávání SME klientů na takové, kteří jsou ve WIND
- Máže osobní údaje, pokud pomine poslední účel použití.
 - "Mazání" klienta v COPRYS a jeho dat řídí příznak "Mazání klienta" se třemi hodnotami
 - Mazat dla MASTER ○ COPRYS klienta "smaže" na základě mazací zprávy ze systému MASTER
 - Mazání blokováno ○ COPRYS klienta nemaže, a pokud by přišla z MASTER mazací zpráva zapamatuje si ji pro pozdější použití
 - Smažat ○ COPRYS klienta "smaže"
 - Při "smazání" klienta musí COPRYS smažat i ty jeho odběratele, kteří nejsou odběrateli jiných klientů
 - Mazání logů aktivit ○ Logy aktivit uživatelů by měly být smazány vždy po 10 letech, žádný log by se neměl uchovávat déle
 - Pokud nastane důvod klienta nebo logy "smažat", COPRYS provede "mazání" v nejbližším běhu jobu PURGER
 - Pro SOFT variantu "mazání" ○ Nová role, která jediná v aplikaci / na GUI uvidí "smazané" klienty a jejich data
- Poskytuje data GDPR aplikaci. ○ COPRYS: Nová funkcionality
- Umí pro určitá data pozastavit jejich odstranění.
- Testování a vývoj NE na produkčních datech

2 varianty "mazání"

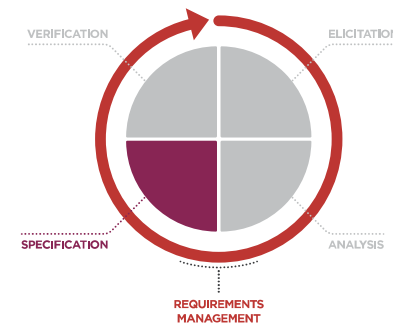
- HARD: "Smažat klienta" = fyzicky jej smažat vč. závislých dat
- SOFT: "Smažat klienta" = zcela jej skrýt, ale ponechat v DB

Shnutí



Requirements Specification

- › Specifikace uživatelských požadavků = písemné zdokumentování “real requirements” tak, jak je v daném kontextu vyžadováno
→ výstup = **documented requirements**
- › Zřídka – izolovaný **katalog funkčních požadavků**
- › Typicky – **funkční specifikace**
 - Strukturovaný dokument doplněný nákresy a diagramy
 - Může obsahovat i seznam uživatelských požadavků
 - Výhoda – uživatelé mu zpravidla dobře rozumí
 - Nevýhoda – při větším rozsahu není snadná aktualizace
 - Ve formátu vhodného nástroje CASE – např. Enterprise Architect
 - Výhoda – exaktní a formalizovaný popis systému
 - Nevýhoda – pro uživatele bývá “suchý a nesrozumitelný”
 - Kombinace obojího
- › Plus **katalog nefunkčních požadavků**
- › Mnohdy také – **model** budoucího systému, tzv. “mockup”
 - Obrazovky, jejich tok, větvení, prokliky - bez funkcionalit a zpravidla bez dat



Req. Specification – doporučení pro požadavky

› Funkční požadavky – formulace

- V požadavku musí být obsažen **aktor, akce** (“capability”)
 - Např. “Uživatel musí zadat heslo a myší stisknout tlačítko Přihlásit”
- V požadavku by měla být obsažena **vymezující kritéria** (“conditions”, “constraints”)
 - Např. “Uživatel musí zadat heslo a do 10 sekund stisknout myší tlačítko Přihlásit”
- Požadavek by měl obsahovat **míru nutnosti** ... “může, musí, měl by”
- Požadavek by měl být formulován **v aktivním rodě** ... tzn. aktor → akce
- Požadavek (zde autor, akce, vymezující kritéria) musí vyjádřen **jednoznačně**
- Požadavek musí (měl by být) **bez gramatických chyb**

› Funkční požadavky – rozšiřující informace

- Požadavek by měl mít **vlastníka** (závisí na kontextu)
- Požadavek musí mít stanovenou **prioritu**, ledaže by byla dána obecně
 - Např. “MoSCoW” – **M**ust have, **S**hould have, **C**ould have, **W**on't have

Req. Specification – doporučení pro požadavky

› Například



Kód	Vlastník	Funkční požadavek	Priorita	Vazby	Reference na FS	Poznámky
FR-4	Core tým	BUS správce katalogu může definovat vlastní služby BRAKE. BUS správce katalogu může definovat nakupované služby a pro ně může definovat: - parametry nákupu služby - parametry třetí strany, která je dodavatelem služby BUS správce katalogu NEmusí v XYZ definovat externí služby	Must - musí		Kap.: 2.2.11	Standardní funkcionalita - bez potřeby dalšího zásahu
FR-5	Core tým	Iniciální portfolio produktů a služeb pokryje svým rozsahem minimálně to-be stav definovaný v SD, a pokryje produktové oblasti "Operativní leasing" a "Fleet management" dokumentované v tabulce 19 v SD.	Must - musí		Kap.: 3.1.5	Zákaznická konfigurace
FR-6	Core tým	Iniciální portfolio produktů a služeb bude nastaveno podle aktuálně používaných parametrů, definovaných v to-be stavu v SD pro komoditu OUA.	Must - musí		Kap.: 3.1.6 - pouze OUA Tabulka 20 + Tabulka 22	Zákaznická konfigurace
FR-7	Core tým	BUS správce XYZ může definovat zákaznické labely.	Could - mohlo by		Kap.: 2.2.4	
FR-8	Core tým	Definované zákaznické labely musí v XYZ být uloženy na hierarchickém principu..	Must - musí	FR-12	Kap.: 2.2.7	
FR-9	Core tým	Oprávněný uživatel, který kalkuluje nabídku, může pro konkrétní nabídku definovat hodnoty k labelům.	Could - mohlo by		Kap.: 2.2.4	
FR-10	Core tým	BUS správce XYZ může definovat zákaznické pohledy.	Could - mohlo by		Kap.: 2.2.5	
FR-11	Core tým	Definované zákaznické pohledy musí být uloženy na hierarchickém principu..	Must - musí	FR-12	Kap.: 2.2.7	
FR-12	Core tým	Kalkulační parametry musí být v XYZ uloženy na hierarchickém principu	Must - musí		Kap.: 2.2.7	
FR-13	Core tým	Kalkulační parametry, které jsou prostou hodnotou nebo jsou seznamem prostých hodnot, musí být v XYZ uloženy formou enumerací.	Should - mělo by		Kap.: 2.2.10	
FR-18	Core tým	BUS správce parametrů XYZ musí definovat alespoň jednoho mandanta; může definovat více než jednoho mandanta.	Won't - nebude		Kap.: 2.3.3	

Req. Specification – doporučení pro požadavky

› Nefunkční požadavky – formulace

- V požadavku musí být uvedena **metrika** (“capability”)
 - Např. “Doba vyhledávání v GUI bez použití našeptávače ...”
- V požadavku musí být uvedena **kritéria** pro danou metriku, pokud možno měřitelná (“conditions”, “constraints”)
 - Např. “... musí být kratší než 30s”
- Požadavek (zde metrika a kritéria) musí vyjádřen **jednoznačně**
- Požadavek musí (měl by být) **bez gramatických chyb**

› Nefunkční požadavky – rozšiřující informace

- Měla by být definována **kategorie** požadavku
 - Např. *Usability, Reliability, Performance, Security, Supportability, ...*
- Požadavek by měl mít **vlastníka** (závisí na kontextu)
- Požadavek musí mít stanovenou **prioritu**, ledaže by byla dána obecně
 - Např. “MoSCoW”

Req. Specification – doporučení pro požadavky

› Například

Kategorie	Kód	Nefunkční požadavek	Priorita	Vlastník
Reliability	NFR001	Databáze aplikace musí být zálohována tak, že po obnovení z případného výpadku budou ztracena maximálně data, která byla uložena 24 hodin nebo kratší dobu před výpadkem.	Must	IT
Performance / Efficiency	NFR002	Doba otevření nové obrazovky nebo přechodu mezi obrazovkami musí být kratší než 2 sekundy	Must	AM
Performance / Efficiency	NFR003	Doba generování tiskového výstupu, který se vztahuje na jeden hlavní datový objekt a má méně než 10 stran musí být kratší než 5 sekund.	Must	AM
Usability	NFR004	Pro vyhledávání indexovaných dat z konkrétních entit musí systém použít našeptávač s funkcí autocomplete	Should	BUS
Performance / Efficiency	NFR007	Doba synchronního generování uživatelsky zaměřeného operativního reportu musí být kratší než 30 sekund	Should	AM
Performance / Efficiency	NFR008	Pro uživatelsky zaměřené operativní reporty, které principiálně trvají déle než 30 sekund, musí být k dispozici možnost asynchronního generování	Must	AM
Supportability / Maintainability	NFR017	Systém musí být provozován v clusteru aplikačních a databázových serverů	Must	AM
Supportability / Maintainability	NFR019	Systém musí auditovat datové změny tak, aby u každé změněné hodnoty a každého vloženého nebo odstraněného záznamu byl dohledatelný původce změny a provedená změna.	Must	AM
Usability	NFR023	Pro exporty a importy dat do/z formátů MS Office musí systém podporovat verze MS Office 2007 a vyšší.	Must	IT
Performance / Efficiency	NFR026	Systém musí umožnit založit a spravovat až 5 000 interních uživatelů	Must	AM
Performance / Efficiency	NFR028	Systém musí umožnit souběžnou práci až 1 000 interních uživatelů v jednu okamžiku	Must	AM
Performance / Efficiency	NFR030	Systém musí umožnit zpracování minimálně 200 databázových transakcí za sekundu	Must	AM

Requirements Specification – nástroje CASE / UML

› CASE / UML

- Prostředek pro reprezentaci vyvíjeného SW na úrovni analýzy, návrhu a zčásti i realizace

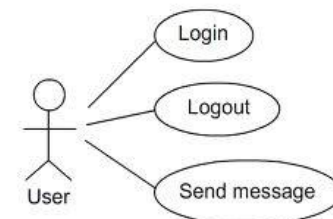


› Proč ano

- Exaktní a formalizovaný popis systému
 - Zatímco textový popis snese (skoro) všechno, strukturovaný vizuální jazyk podporovaný logikou nástroje CASE (zpravidla) ne
- Definované konceptuální pohledy s vazbou na process vývoje, např. obrazovky a jejich toky, vstupy / výstupy, stavové diagramy, aktivity diagramy, rozhraní, E-R diagramy (DB)

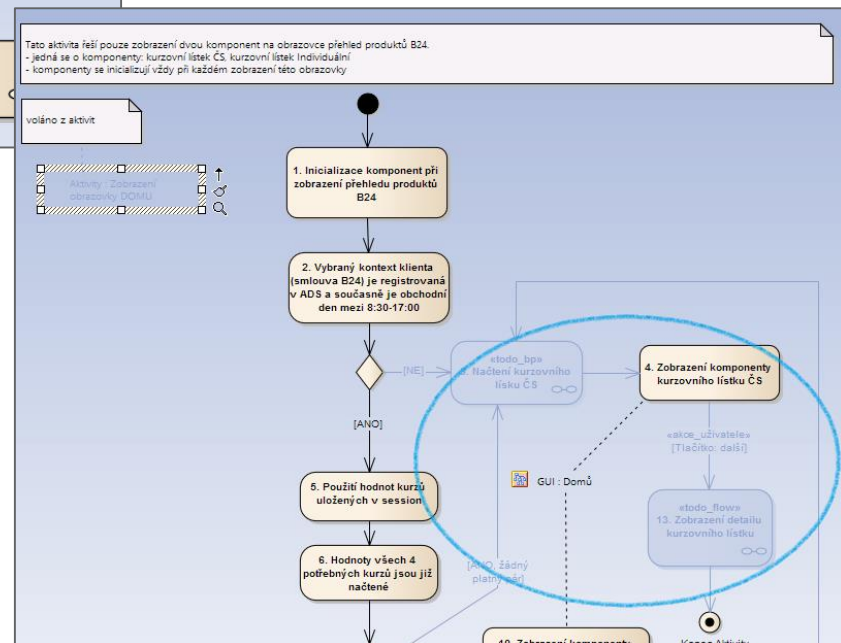
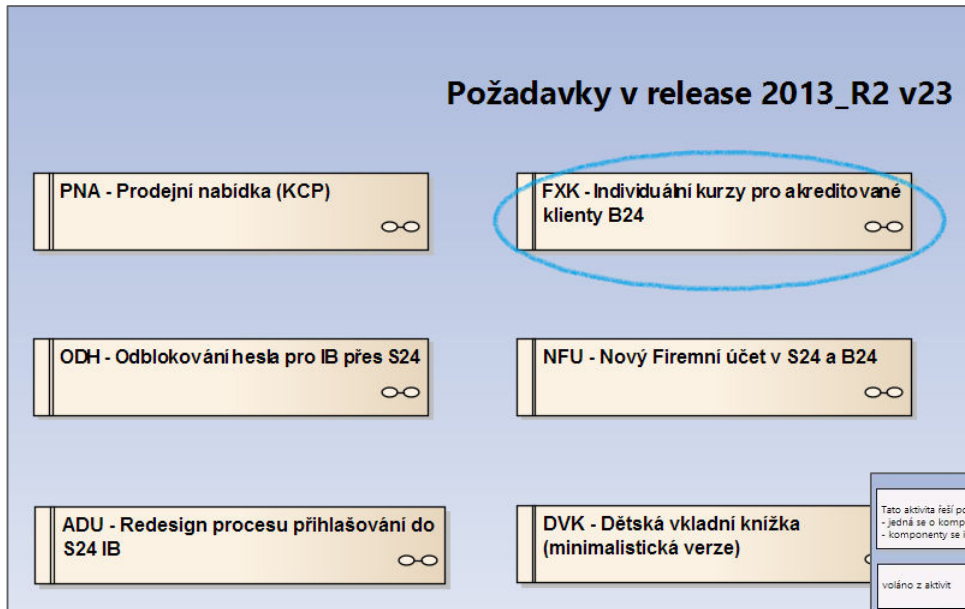
› Ale...

- Obtížné zachycení obchodních požadavků, celkových souvislostí, “big picture”
- Pro zadavatele mnohdy “suché a nesrozumitelné”
- Pasti
 - Svádí k fragmentovanému pohledu bez hlubších souvislostí
 - Use cases
 - Zadavatel jim rozumí, ale bývají přeceňovány
 - Jako součást specifikace ano, jako základ specifikace ne (viz výše “User profiling”)



Requirements Specification – nástroje CASE / UML

› Příklady



Req. Specification – doporučení pro specifikaci ... dle IEEE

- › Correct
- › Unambiguous
- › Complete
- › Consistent
- › Ranked
- › Verifiable
- › Modifiable
- › Traceable

1. **Correct:** The SRS, or software requirements specification, should correctly describe the system behavior. It is not productive to have a requirements document that describes implausible or impossible expected system behavior or user goals.

2. **Unambiguous:** Software requirements should be written in such a manner as they are not subject to different interpretations. The use of specific and appropriate language can help avoid ambiguity in interpretation.

3. **Complete:** the software requirements document should completely describe the system's expected behaviors and feature set.

4. **Consistent:** Requirements for the system under discussion must not contradict each other.

5. **Ranked:** You must rank your software requirements for importance. Each software requirement has its own level of importance and criticality, and they are not all equal. By ranking the requirements, software designers ensure that guidance is given to the development team regarding effective prioritization.

6. **Verifiable:** If the requirement cannot be verified as having been met, then the requirement itself is written poorly. The requirements have to be testable.

7. **Modifiable:** The requirements must be easy to modify or change.

8. **Traceable:** The requirements must be traceable, and it is essential that traceability information has been provided, as the requirements document provides the starting point in the traceability chain. I have written elsewhere in this blog at length about the importance of software requirements traceability and have provided examples of software requirement traceability matrixes. Many software development organizations use proprietary CASE software tools and other methods to enforce traceability policies that stipulate how much traceability information regarding requirements must be maintained.

Requirements Specification – trasovatelnost



- › Dříve či později vyvstanou v projektu dotazy
 - Jaký je účel tohoto požadavku? Kdo jej zadal?
 - Proč je v systému tato funkce?
 - Pokrývá navžené řešení všechny požadavky?
 - Jaké dopady bude mít změna požadavku XY?
 - Jsou pro všechny požadavky testovací scénáře?
 - ...
- Je žádoucí znát vazby mezi požadavky.

Traceability is the degree to which a relationship can be established between two or more products of the requirement engineering process.

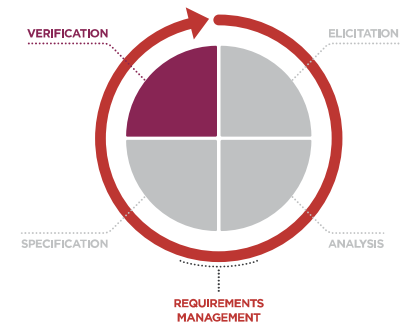
Institute of Electrical and Electronic Engineers (IEEE), www.ieee.org

- › Vytváření a udržování vazeb mezi požadavky (zpravidla různých úrovní) je nedílnou součástí životního cyklu vývoje software (SDLC), začíná v rámci “Requirements Engineering”

Requirements Verification / Validation

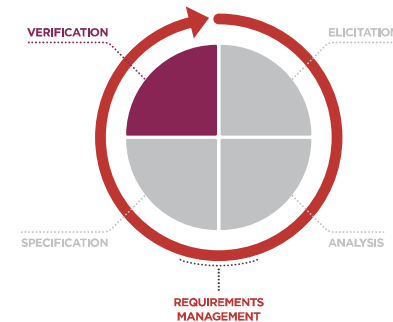
Jsou specifikované požadavky...

- › Věcně správné?
 - Jsou požadavky úplné? Jsou zahrnuty i ty “samozřejmé”?
 - Popisují požadavky to, co je opravdu potřeba? Gold-plating?
- › Proveditelné?
 - Jsou specifikované požadavky realizovatelné (technologie, bezpečnost, regulatorní omezení, ...)?
- › Testovatelné?
 - Bude možné otestovat, že jsou požadavky splněny?
- › Trasovatelné?
 - Jsou mezi požadavky (různých úrovní) vytvořeny vazby?
- › Formálně správné?
 - Jsou požadavky specifikovány v souladu s doporučeními a s nároky daného kontextu?
- › Výstup → **verified/valid requirements**



→ Cíl = minimalizovat pravděpodobnost “překvapení”

Requirements Verification / Validation

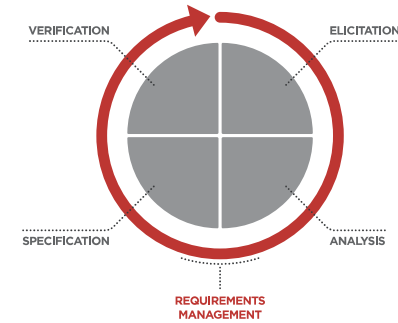


Ověření věcné správnosti

- › Workshop “Představení požadavků”
 - Účastní se “stakeholders”
 - Vede / moderuje autor dokumentovaných požadavků
 - Cíl – strukturovaně představit a sesumarizovat dokumentované požadavky
- › Připomínkování požadavků
 - “Stakeholders” obdrží dokumentované požadavky s cílem je prostudovat a napsat připomínky
 - Best practice – připomínky zaznamenávat formou katalogu (trasovatelnost!) → tabulka ve sdíleném spreadsheetu apod.

Id	Kapitola	Připomínka	Od koho	Zpracováno ano/ne	Způsob vyřešení připomínky
1	2.1 Business zadání	Rodné číslo je buďto vyplnění a pak se validuje, nebo prázdné a nevaliduje se. ICO validace jsou tímto nedotčeny.	ABR	ano	Text zadání upraven
2	2.2 Kontext	Doplnit do DQ kontrolu délky polí	ABR	ne	Je popsáno v kapitole 5 DQ kontroly pohledávek
3	3.1 Nahrání pohledávek - požadovaný stav	Odstranit datum narození	VHR	ne	Text je již označen šedě, tj. mimo scope; zbytek dokumentu prohledán a ostatní místa s datem narození vyšeděna.
4		Požadujeme stav, kdy hodnotíme r.č., jeli vyplněno, jinak může být nevyplněné. ICO beze změn	ABR	ne	Je popsáno v kapitole 5 DQ kontroly pohledávek
5		Není doplněna DQ kontrola na délku polí, pokud se má implementovat	ABR	ne	Je popsáno v kapitole 5 DQ kontroly pohledávek
6	4.1 Datové opravy pohledávek - požadovaný stav	Vyřadit z textu datum narození	FSO	ne	Text je označen šedě, tj. mimo scope
7		Zažlutit dolní iČO zleva na 8 znaků	ABR	ano	Zažluceno
8		Kód banky bude součástí čísla účtu vždy, stávající nepropojení považují za chybu	ABR	ne	Vstupní data jsou dodávána i v nepropojeném stavu - viz [Pohl_úcty_vzorek] a vpravo uvedený příklad z pohledávek
9		DOTAZ: IČO/RČ/ZIČ - ponechat ještě lomítka?	PMI	ano	Rozhodnuto na následném WS: Lomítka také vyhodit
10		DOTAZ: Chceme v rámci datových oprav filtrovat dvojité uvozovky? ... INCS168076	PMI	ano	Rozhodnuto na následném WS: ANO dvojité uvozovky filtrovat ze všech polí pohledávek
11		DOTAZ: "V případě, že všechna políčka určující sídlo odběratele jsou prázdná, je možné za předpokladu, že kód země je CZ nebo SK dotáhnout hodnoty z RES/ORSR" ... platí ještě toto zadání?	PMI	ano	Rozhodnuto na následném WS: Vykomentovat z kódu nebo v kódu jinak zaslepit

Requirements Management



- › V širším slova smyslu se “management” týká celého procesu Requirements Engineering

- › **Action planning** – plánování aktivit
 - Mýty
 - Proces specifikace požadavků nelze plánovat
 - Proces specifikace požadavků lze přesně plánovat
 - Skutečnost
 - Proces specifikace požadavků lze plánovat, avšak jen do určité míry
 - Typicky není zcela jasný rozsah a často je třeba stavět na předpokladech

- › **Managing changes** – správa změn
 - Ve všech fázích procesu může docházet ke změnám požadavků – a dochází
 - Možnost reagovat na změny se v průběhu procesu zmenšuje
 - Nepřímo úměrně tomu roste potřeba formalizovat zacházení se změnami
 - Každá změna nemusí být “change” – pouze větší míra detailu?
 - “Boj” o rozsah

Shrnutí a postřehy z praxe

Analýza v agilní organizaci

~~Big Design Up Front~~

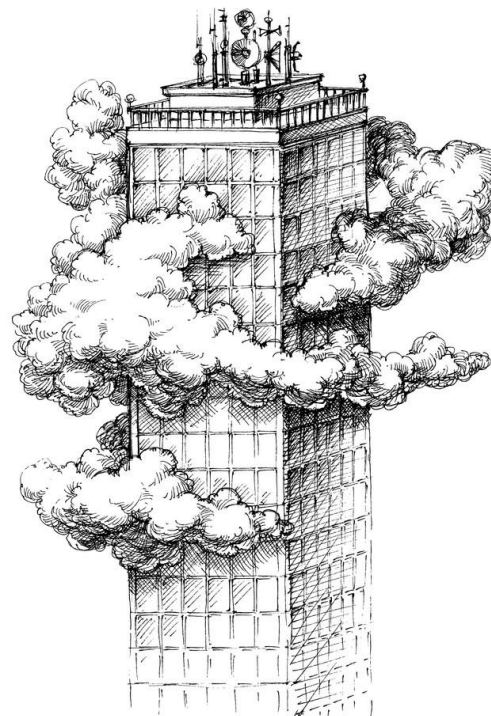


„Sufficient design“
„Emergent design“

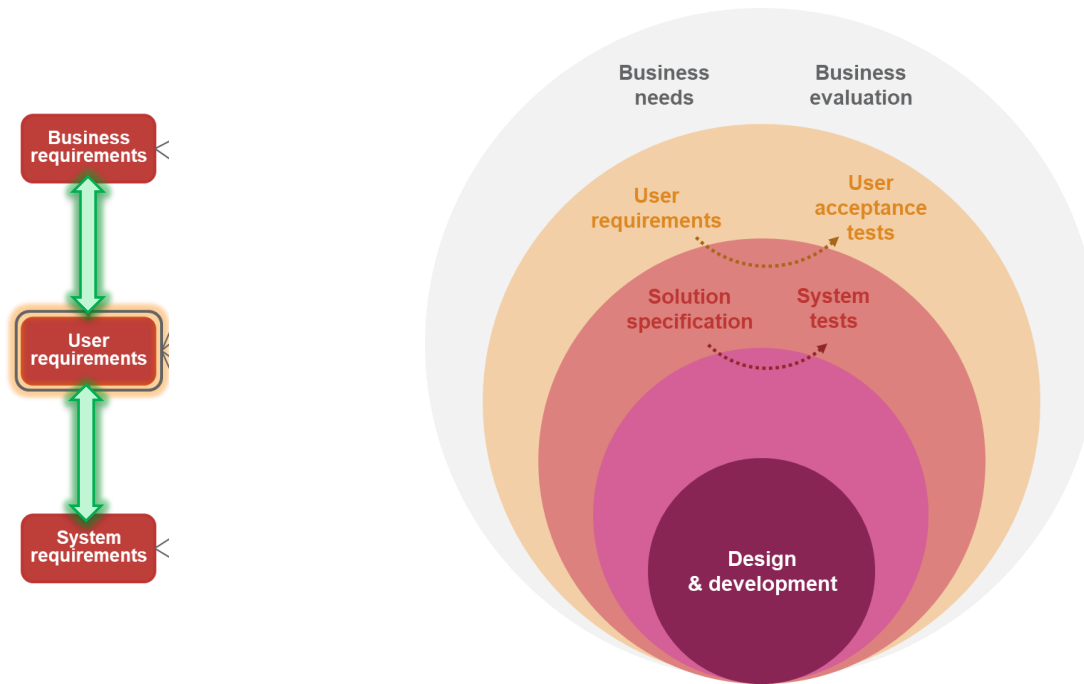
ale přesto

“Start with the End in Mind”

Aneb je třeba mít alespoň v hrubých obrysech jasno, jaký je cílový stav.
→ Vyvarujme se “intelektuálního dluhu”.



Requirements Engineering – fixace rozsahu



FIXACE ROZSAHU

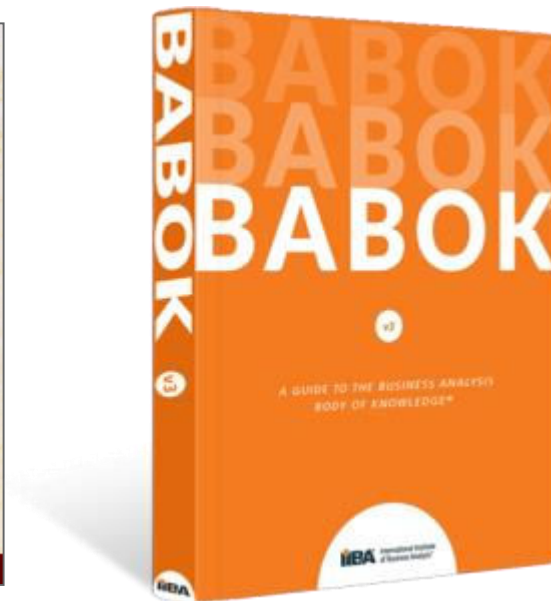
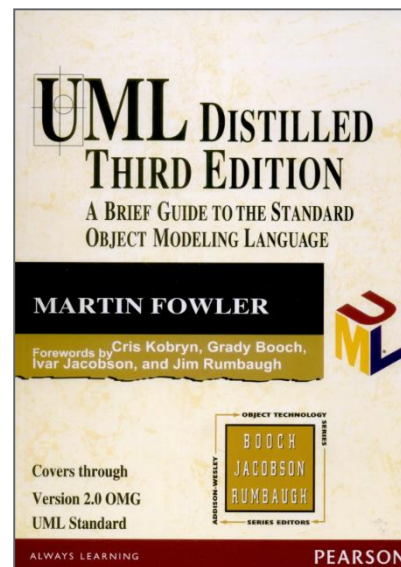
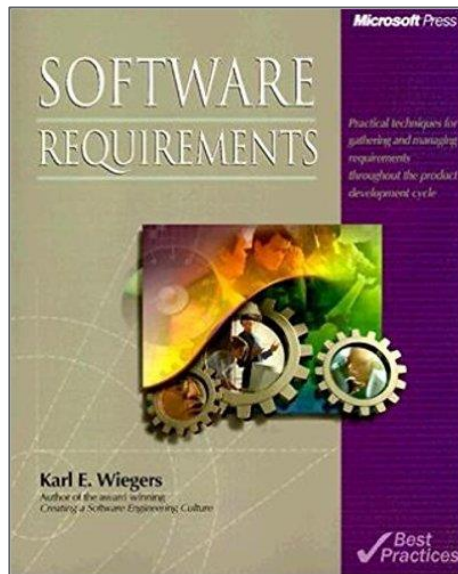
- › Pro projekty zásadně důležitá
- › Nejčastější netriviální příčina neúspěchu
- › I přes úsilí bývá “šedá zóna”

Rady na závěr

- › Vyvaruj se školáckých chyb, např.
 - Ignorování důležitého stakeholdera, nefunkčních požadavků, věcí, které nechci slyšet, “nevím, co nevím” , architektury
- › Vyvaruj se intelektuálního dluhu
 - Pokus se odolat tlaku dodat „rychlou byznysovou hodnotu“, není-li aspoň v obrysech znám konečný cíl
 - Než začneš s detaily, pokus se dohlédnout na konec
 - Vydrž to, že “nevíš”, a nepokoušej se hned hledat řešení
- › Přizpůsob výrazové prostředky “publiku”
 - Sebedokonalejší analýza nemá cenu, pokud jí nebude rozumět zadavatel a/nebo designer/vývojář
- › Měj na paměti rozsah

Další studium

- › **K. Wiegers: Software requirements**
 - Zlatý standard
- › **M. Fowler: UML Distilled**
 - Praktické, pragmatické a čtivé
- › **BABOK**
 - Pokročilá témata



Templates, checklists, literatura

Materiály SWENG - Requirements

ČLÁNKY

- ▶ [When Telepathy Won't Do: Requirements Engineering Key Practices](#)
- ▶ [Karl Wiegers Describes 10 Requirements Traps to Avoid](#)
- ▶ [Writing Effective Natural Language Requirements Specifications](#)
- ▶ [Be Careful With "Use Cases"](#)

KNIHY

- ▶ Wieger K. Software Requirements: Practical techniques for gathering and managing requirements throughout the product development cycle. Microsoft Press, 1999. resp. 2nd ed.
- ▶ [Just Enough Structured Analysis](#) - klasická kniha Ed Yourdona ve wiki formátu !

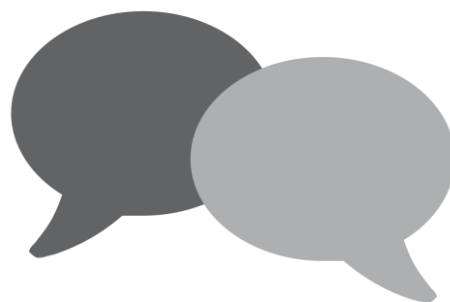
CHECKLISTS

- ▶ [CxCheck_Requirementstxt](#) - checklist firmy Construx pro oblast Requirements
- ▶ [Requirements_review_checklist.doc](#)
- ▶ [Use_case_checklist.doc](#)
- ▶ [Impact_analysis_checklist.doc](#)

TEMPLATES

- ▶ [srs_template.doc](#) - velmi dobrý template pro tvorbu specifikace požadavků
- ▶ [use_case_template.doc](#)
- ▶ [vision_and_scope_template.doc](#)
- ▶ [SAFE_BusinessRequirements.doc](#)
- ▶ [SAFE_SystemRequirements.doc](#)

Všechny odkazované materiály jsou poskytnuty výhradně za účelem výuky softwarového inženýrství.



Dotazy?

Děkuji za pozornost

PROFINIT

NÁSKOK DÍKY ZNALOSTEM

Profinit EU, s.r.o.

Tychonova 2, 160 00 Praha 6 | Telefon + 420 224 316 016



Web
www.profinit.eu



LinkedIn
linkedin.com/company/profinit



Twitter
twitter.com/Profinit_EU



Facebook
facebook.com/Profinit.EU



Youtube
[Profinit EU](https://Profinit.EU)