http://www2.fiit.stuba.sk/~lang/PSI/

Princípy softvérového inžinierstva

doc. Ing. Ján Lang, PhD.

Ústav informatiky, informačných systémov a softvérového inžinierstva (UISI)

Fakulta informatiky a informačných technológií

Slovenská technická univerzita v Bratislave

Ilkovičova 2, 84216 Bratislava 4

jan.lang@stuba.sk

19.02.2026

S T U

F I I T

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

Faculty of Informatics and Information Technologies

Úvod a organizácia predmetu

Počet kreditov: 6

Prednášky: doc. Ing. Ján Lang, PhD., pozvaní hostia

Cvičenia:

- Ing. Bystrík Bindas
- Ing. Ivan Kapustík
- Ing. Vladimír Mlynarovič, PhD.

Jazyk výučby: slovenčina

Celkové hodnotenie

1. Hodnotenie v priebehu obdobia výučby predstavuje **70%** celkového hodnotenia
2. Záverečné hodnotenie je realizované písomnou skúškou, ktorá predstavuje **30%**

1. Hodnotenie v priebehu obdobia výučby

Projekt (max 55 b)

Aktivita v predmete (max 15 b)

Projekt

Cieľom semestrálneho projektu je oboznámiť sa s podstatnými fázami realizácie projektu zameraného na tvorbu modelu softvérového systému vrátane jeho vykonateľnej podoby.

Projekt - hlavné časti

1. Zameranie projektu (5 b)
2. Špecifikácia požiadaviek, prípady použitia (15 b)
3. Iniciálny model správania v UML (10 b)
4. Štruktúra systému, objekty, triedy a interakcia (15 b)
5. Implementácia, kód (10 b)
6. Finálna správa o riešení projektu

Projekt - termíny odovzdania

1. Zameranie projektu, **do 02.03.2026 08:00**
2. Špecifikácia požiadaviek, prípady použitia, **do 16.03.2026 08:00**
3. Iniciálny model správania v UML, **do 07.04.2026 08:00**
4. Štruktúra systému, objekty, triedy a interakcia, **do 27.04.2026 08:00**
5. Implementácia, kód, **do 04.05.2026 08:00**
6. Finálna správa o riešení projektu, posledný deň výučby v letnom semestri **do 15.05.2026 23:59**

Projekt - odovzdania

Všetky modely musia byť odovzdané ako **jeden EA projekt**.

Textové dokumenty sa odovzdávajú v **PDF**.

Implementácia sa odovzdáva v jednom **zip** archíve.

Všetky odovzdania sú realizované prostredníctvom **AIS**.

Projekt - odporúčané nástroje, technológie

1. Zameranie projektu (**LaTeX, MS Word**)
2. Špecifikácia požiadaviek, prípady použitia (**LaTeX, MS Word**)
3. Iniciálny model správania v UML (**EA, LaTeX, MS Word**)
4. Štruktúra systému, objekty, triedy a interakcia (**EA, LaTeX, MS Word**)
5. Implementácia, kód (**IntelliJ, GitHub, Jira, LaTeX, MS Word, OOP jazyk: Java, C++, C#, Python)**
6. Finálna správa o riešení projektu (**LaTeX, MS Word, GitHub, Jira, Figma**)

Projekt - 1. Zameranie projektu

Vypracujte zámer projektu ako spresnenie rámcovej témy tak, aby bolo jasné, aký softvérový systém budete modelovať.

Rámcová téma: **Produkty, tovary a služby**

Pod zámerom sa rozumie textový opis systému. Očakávaný rozsah textu zámeru projektu je **cca 2 000 znakov**.

Projekt výstižne **pomenujte**.

K zámeru projektu je **každý člen** tímu povinný vypracovať samostatný **opis jedného prípadu použitia**.

Zámer musí byť **odsúhlasený na cvičeniach**, môže byť potrebná úprava podľa pokynov cvičiaceho.

Ako bude typicky vyzerať práca na projekte

Písanie (štruktúrované aj neštrukturované, US, UC, zdrojak)

Práca s nástrojmi Enterprise Architect (EA) (kreslenie diagramov, písanie textov, vypĺňanie formulárov,...)

Konzultovanie na cvičeniach (cvičiaci je aj váš zákazník, cvičiaci je aj váš mentor)

2. Záverečné hodnotenie - Skúška

Písomná skúška tvorí 30% celkového hodnotenia predmetu.

Na úspešné ukončenie predmetu je potrebné získať minimálne 7 bodov z maximálneho 30 bodového hodnotenia skúšky.

Cvičenia

Plán cvičení dostupný na webe predmetu: https://www2.fiit.stuba.sk/~lang/PSI/cvicenia/

Explicitne vymenované aktivity na cvičenie a príprava na ďalšie cvičenie.

Priebežná kontrola študijných výsledkov (priebežné hodnotenie projektu, kontrolné otázky a pod.).

Cvičenia - očakávania

Aktívny prístup v rámci vlastného projektu.

Aktívna a odborne relevantná medzitímová účasť v diskusiach.

Vyriešenie zadaných úloh, uplatnenie vedomostí z prednášok.

Priebežná prezentácia jednotlivých aj rozpracovaných časti projektu .

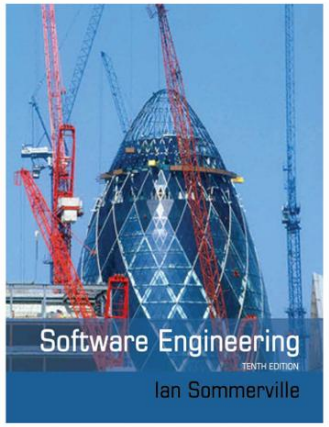
Prezentácia jednotlivých časti projektu po ich odovzdaní.

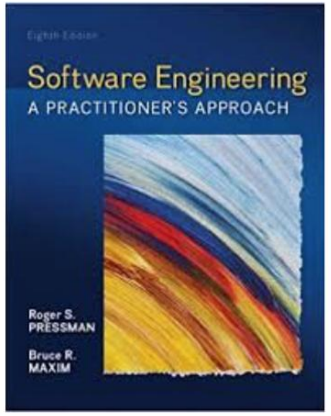
Povinná účasť.

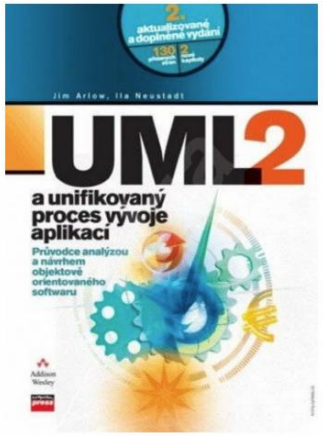
Prednášky

[19. február]: **Úvod do softvérového inžinierstva**
[26. február]: **Softvérové procesy**
[5. marec]: **Analýza a špecifikácia požiadaviek**
[12. marec]: **Princípy návrhu softvérových systémov**
[19. marec]: **Návrh používateľského rozhrania**
[2. apríl]: **Metódy vývoja softvéru**
[9. apríl]: **Implementácia softvérových systémov**
[16. apríl]: **Testovanie softvérových systémov**
[23. apríl]: **Prevádzka a údržba softvérových systémov**
[30. apríl]: **DXC**
[7. máj]: **JetBrains**
[14. máj]: **Konzultácie k projektu**

Literatúra







UNIFIED MODELING LANGUAGE®

OMG®
OBJECT MANAGEMENT GROUP®

[UML diagrams]

[Cockburn - Use Case Book (draft)]

V. Vranić - Modelovanie softvéru (PDF)

[UML špecifikácia (OMG)]

AKO BYŤ ÚSPEŠNÝ?

- Priebežne čítať (zdroje web predmetu, prednášky, cvičenia, literatúra,...)
- Prednášky (aktívna účasť, písať si poznámky, rozmýšľať v súvislostiach, projekt, pýtať sa, pýtať sa, pýtať sa,...)
- Cvičenia (aktívna účasť, konzultovať priebežné výsledky projektu (nevyhnutná príprava), v termíne odovzdať výstupy, projek)

http://www2.fiit.stuba.sk/~lang/PSI/

Úvod do softvérového inžinierstva

doc. Ing. Ján Lang, PhD.

Ústav informatiky, informačných systémov a softvérového inžinierstva (UISI)
Fakulta informatiky a informačných technológií
Slovenská technická univerzita v Bratislave
Ilkovičova 2, 84216 Bratislava 4
jan.lang@stuba.sk

STU FIIT
SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
Faculty of Informatics and Information Technologies

Úvod do softvérového inžinierstva

- Čo je to softvér?
- Čo je to softvérové inžinierstvo?
- Čo sú to princípy?

Úvod do softvérového inžinierstva

- Čo je to softvér? Programy, dáta a dokumentácia. Softvér ako špecifikácia správania systému.
- Čo je to softvérové inžinierstvo?

- Čo sú to princípy?

Úvod do softvérového inžinierstva

- Čo je to softvér? Programy, dáta a dokumentácia. Softvér ako špecifikácia správania systému.
- Čo je to softvérové inžinierstvo? Systematický a inžiniersky prístup k vývoju softvéru.
- Čo sú to princípy?

Úvod do softvérového inžinierstva

- Čo je to softvér? Programy, dáta a dokumentácia. Softvér ako špecifikácia správania systému.
- Čo je to softvérové inžinierstvo? Systematický a inžiniersky prístup k vývoju softvéru.
- Čo sú to princípy? Základné pravidlá, zásady alebo myšlienky, ktoré usmerňujú, ako niečo robiť správne, efektívne a konzistentne.

Úvod do softvérového inžinierstva

Prečo je softvérové inžinierstvo potrebné?

Koľko percent softvérových projektov skončí úspešne?

slido

Úvod do softvérového inžinierstva

Koľko percent softvérových projektov skončí úspešne? 0

- <30%
- niečo medzi 30% - 40%
- niečo medzi 40% - 60%
- niečo medzi 60% - 70%
- >70%

Úvod do softvérového inžinierstva

Koľko percent softvérových projektov skončí úspešne? 0

- <30%
- niečo medzi 30% - 40%
- niečo medzi 40% - 60%
- niečo medzi 60% - 70%
- >70%

Standish Group - CHAOS Report on IT Project Outcomes, https://www.standishgroup.com/

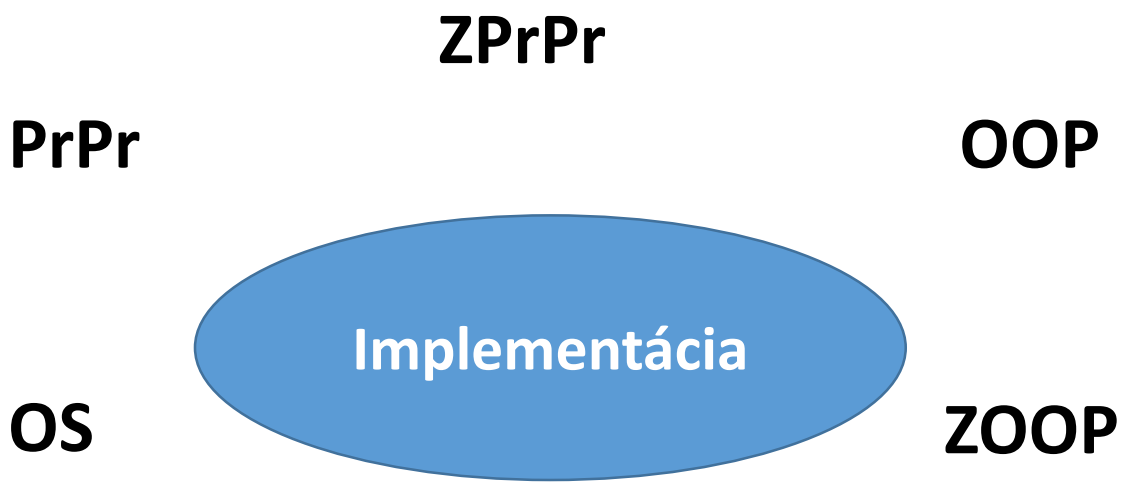
Prečo je softvérové inžinierstvo stále potrebné?

...nestačí podpora AI?

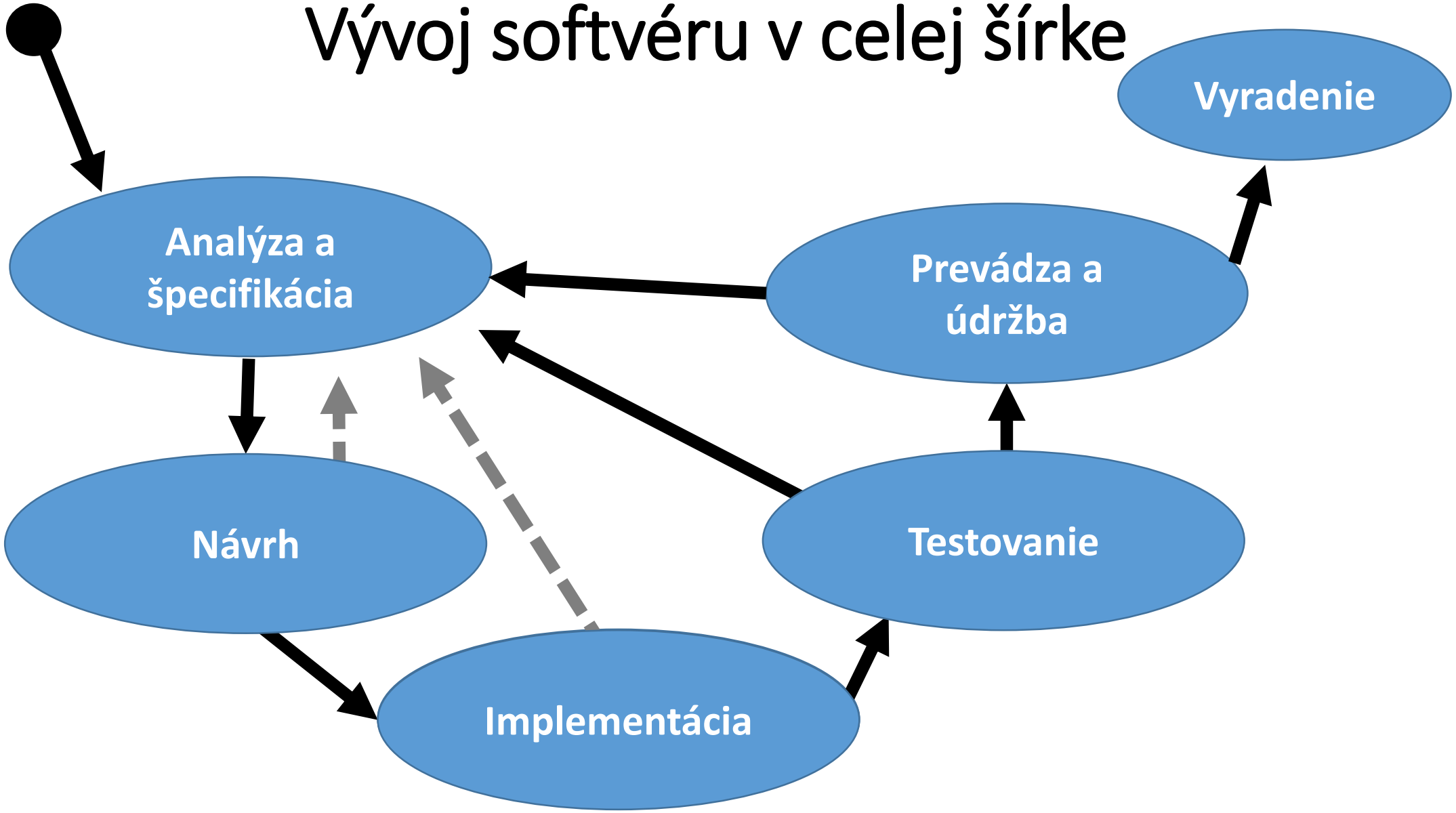
Prečo je softvérové inžinierstvo stále potrebné?

...nestačí podpora AI?

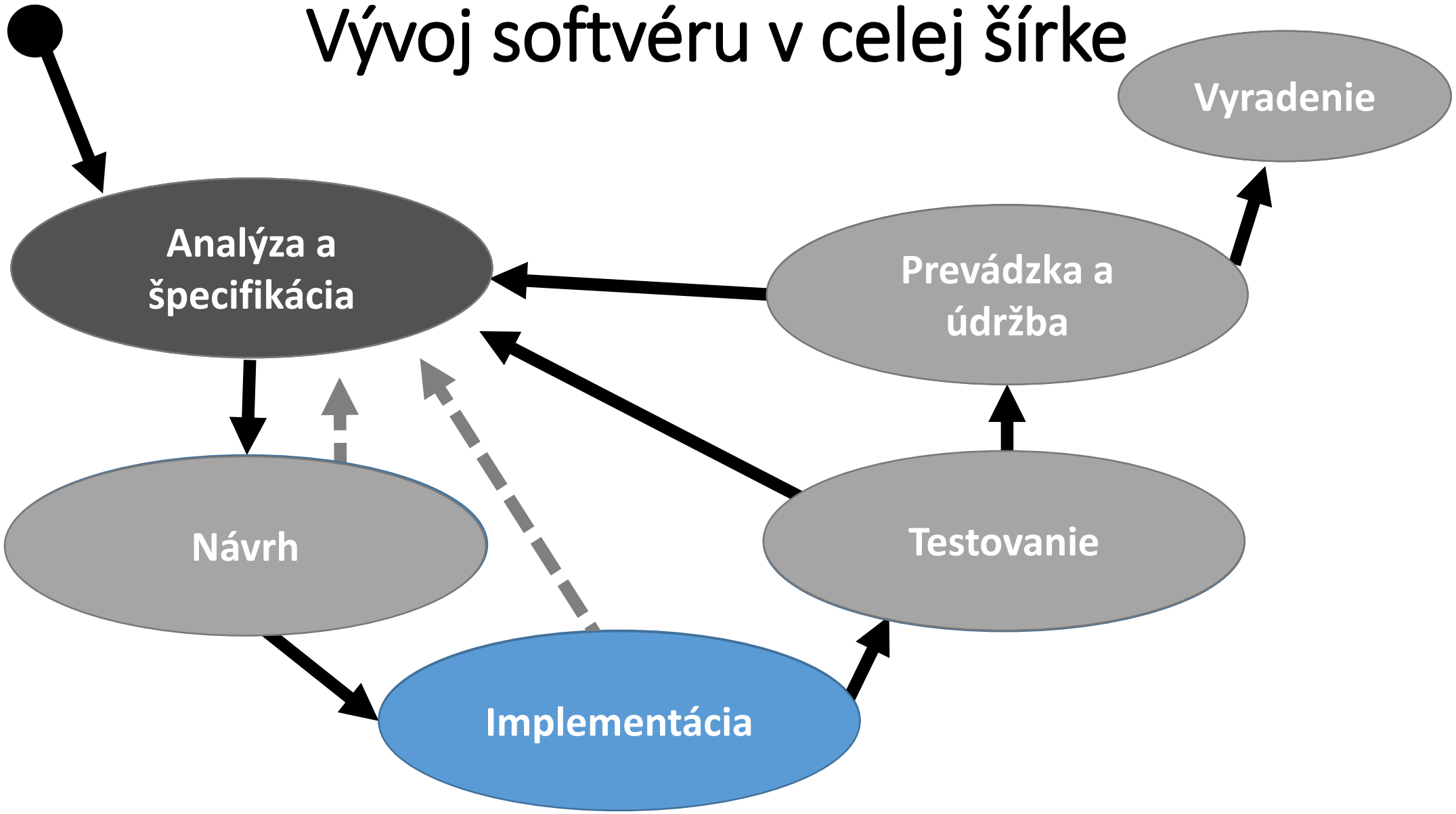
Máte za sebou predmety, ktoré viac menej korelujú k spoločnému...



Vývoj softvéru v celej šírke



Vývoj softvéru v celej šírke



Vývoj softvéru v celej šírke

Už ste nejaký veľký softvér vytvorili?

Kedy je už softvér veľký? (dajte príklad)

V čom je vlastne problém pri tvorbe veľkých projektov/softvérov?

Vývoj softvéru v celej šírke

Už ste nejaký veľký softvér vytvorili?

Kedy je už softvér veľký? (dajte príklad)

V čom je vlastne problém pri tvorbe veľkých projektov/softvérov?

- Rastúca veľkosť a komplexnosť systémov. Začína byť problém všetko si pamätať
- Tímová spolupráca. Jeden človek nezvládne všetky špecializácie
- Dlhodobý vývoj a meniace sa požiadavky. Fluktuácia je realita

Vývoj softvéru v celej šírke

Životný cyklus softvéru:

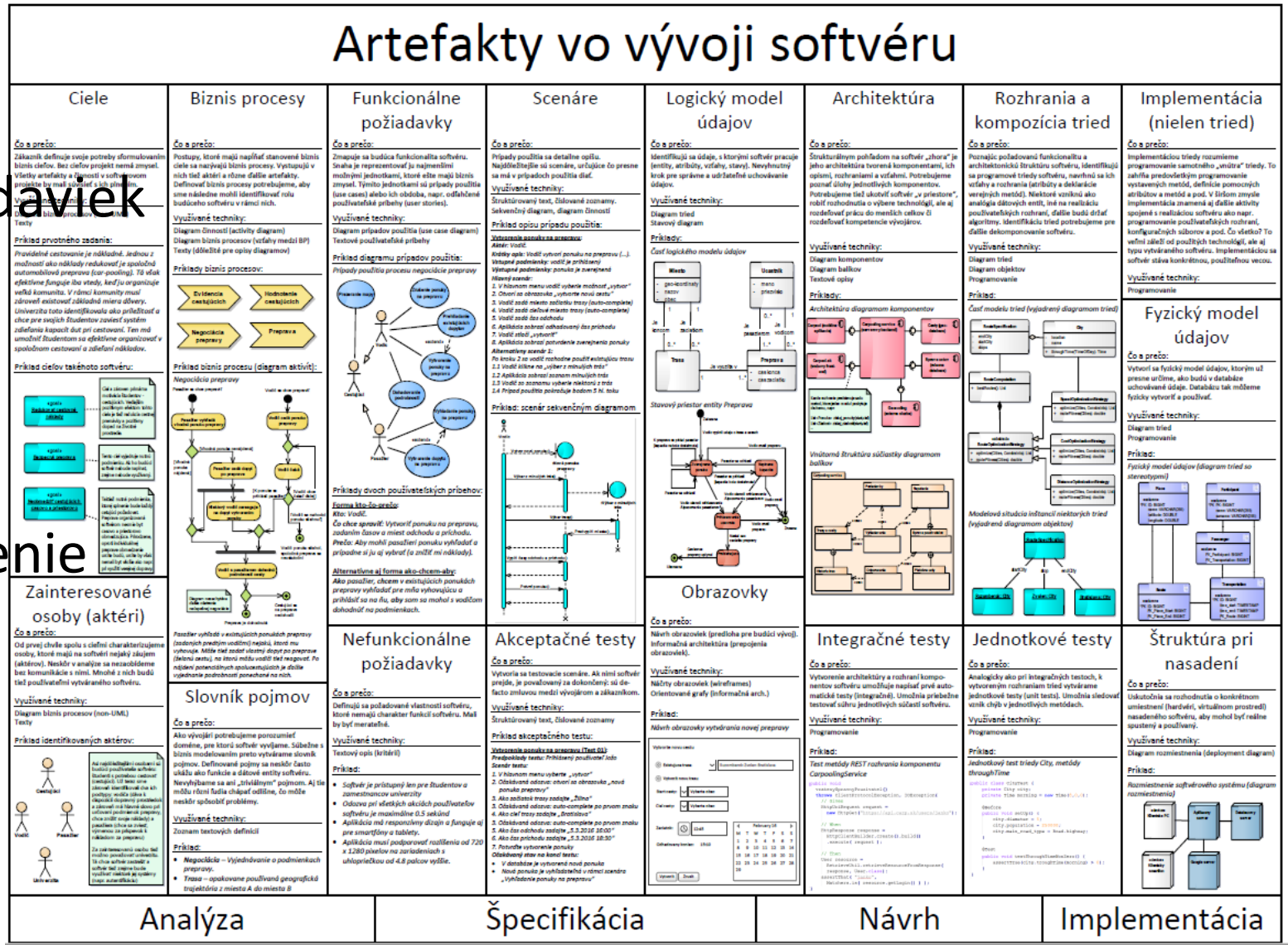
Analýza, špecifikácia požiadaviek

Návrh systému

Implementácia

Testovanie

Prevádzka a údržba, vyradenie



Analýza, špecifikácia požiadaviek

Aky je problem?

Zlé požiadavky – zlé odhady – zlý plan – predraženy softvér, ktorý robí niečo iné ako chceme

Have you any experience in customer requirements elicitation

?

Have you any experience in customer requirements elicitation?

...e.g. in general, even specific related to softvare being developed?

How did you/would you record those requirements?

How did you/would you record those requirements?

...how it may look like? Based on these requirement engineers will develop the system...

...an example:

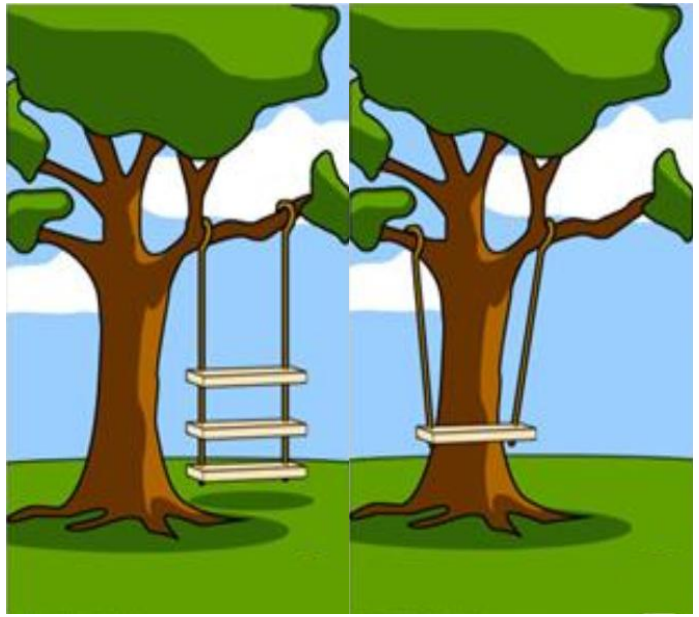
What the customer wanted for a swing and what transpires throughout the software (product) development lifecycle

Tree Swing spec & after

How the customer explained it	How the project leader understood it	How the analyst designed it	How the programmer wrote it	What the beta testers received

https://www.zentao.pm/blog/tree-swing-project-management-tire-analogy-426.html

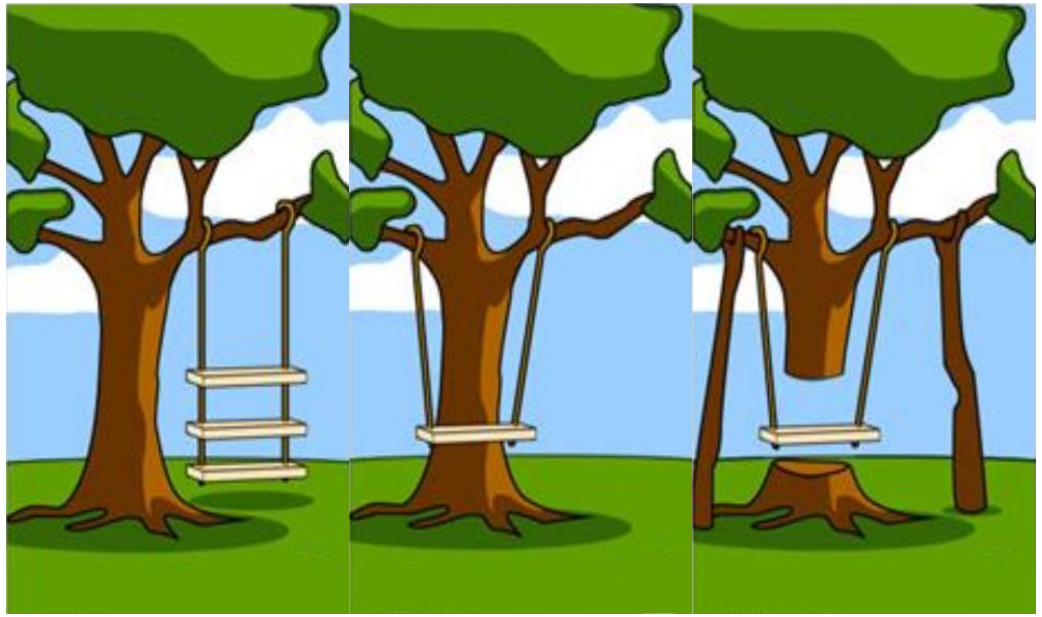
Tree Swing spec & after



How the customer explained it	How the project leader understood it	How the analyst designed it	How the programmer wrote it	What the beta testers received

https://www.zentao.pm/blog/tree-swing-project-management-tire-analogy-426.html

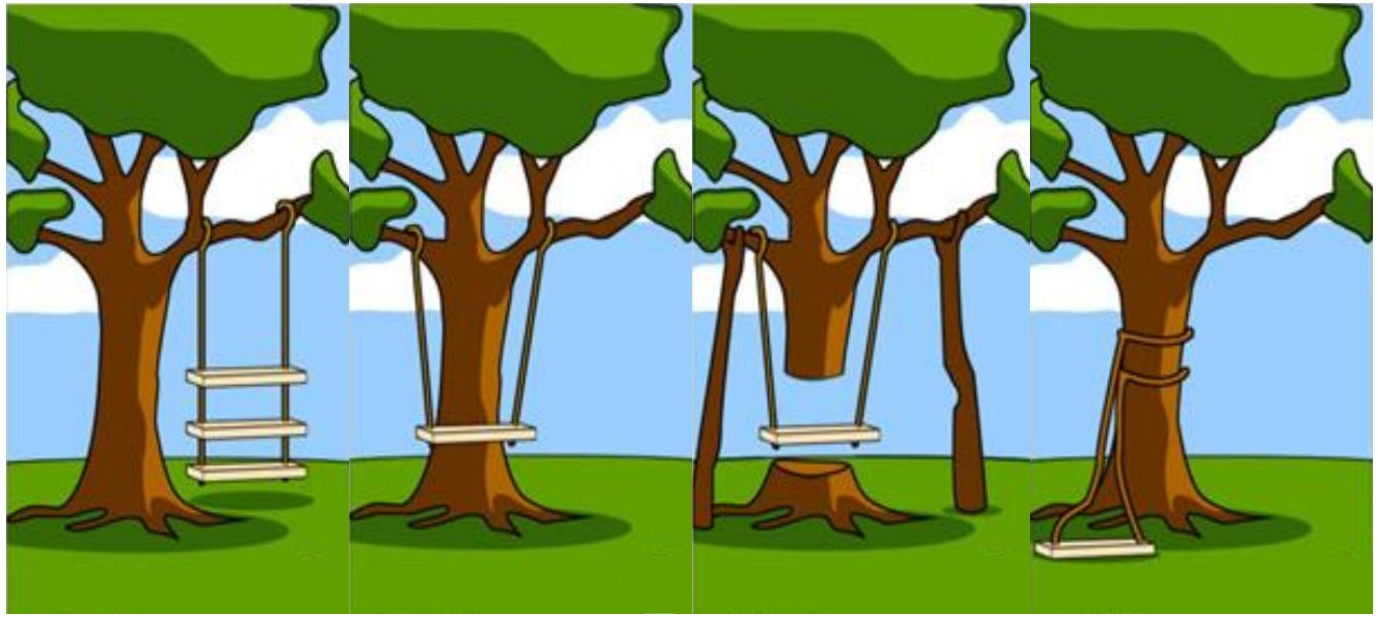
Tree Swing spec & after



How the customer explained it	How the project leader understood it	How the analyst designed it	How the programmer wrote it	What the beta testers received

https://www.zentao.pm/blog/tree-swing-project-management-tire-analogy-426.html

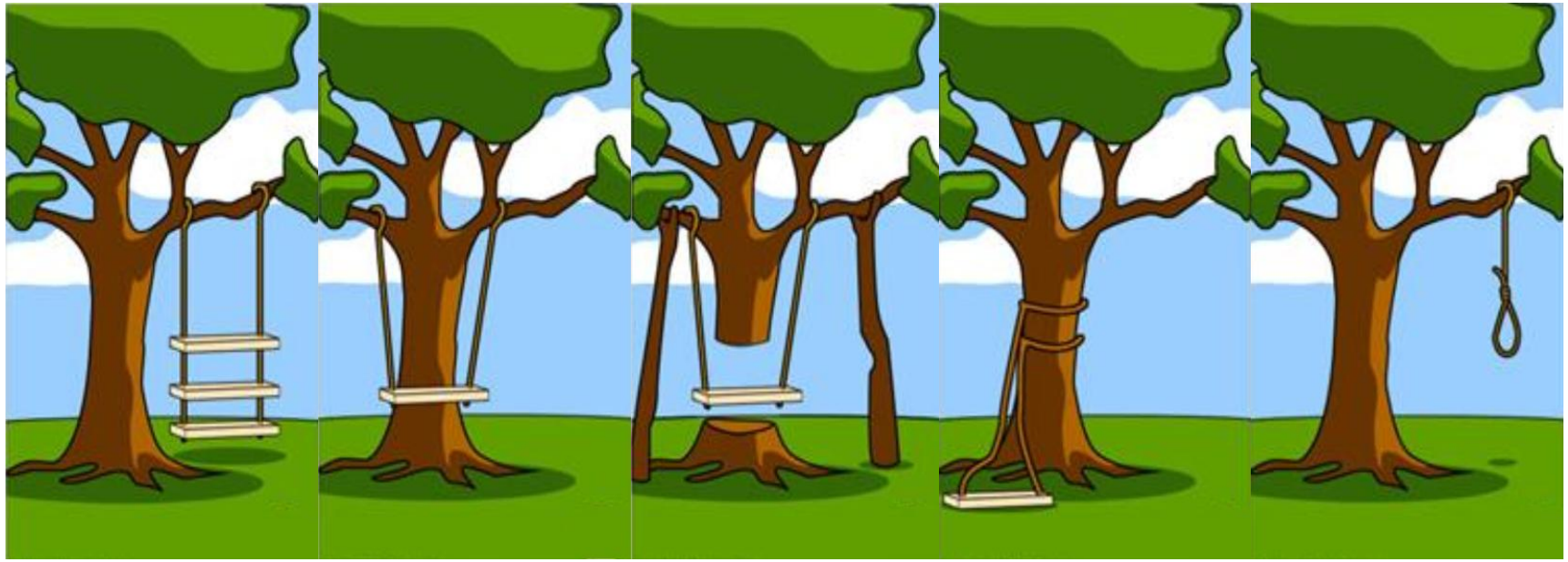
Tree Swing spec & after



How the customer explained it	How the project leader understood it	How the analyst designed it	How the programmer wrote it	What the beta testers received

https://www.zentao.pm/blog/tree-swing-project-management-tire-analogy-426.html

Tree Swing spec & after



How the customer explained it	How the project leader understood it	How the analyst designed it	How the programmer wrote it	What the beta testers received

https://www.zentao.pm/blog/tree-swing-project-management-tire-analogy-426.html

Tree Swing spec & after

How the customer explained it	How the business consultant described it	How the project was documented	How the customer was billed	What the customer really needed

https://www.zentao.pm/blog/tree-swing-project-management-tire-analogy-426.html

Tree Swing spec & after

How the customer explained it	How the business consultant described it	How the project was documented	How the customer was billed	What the customer really needed

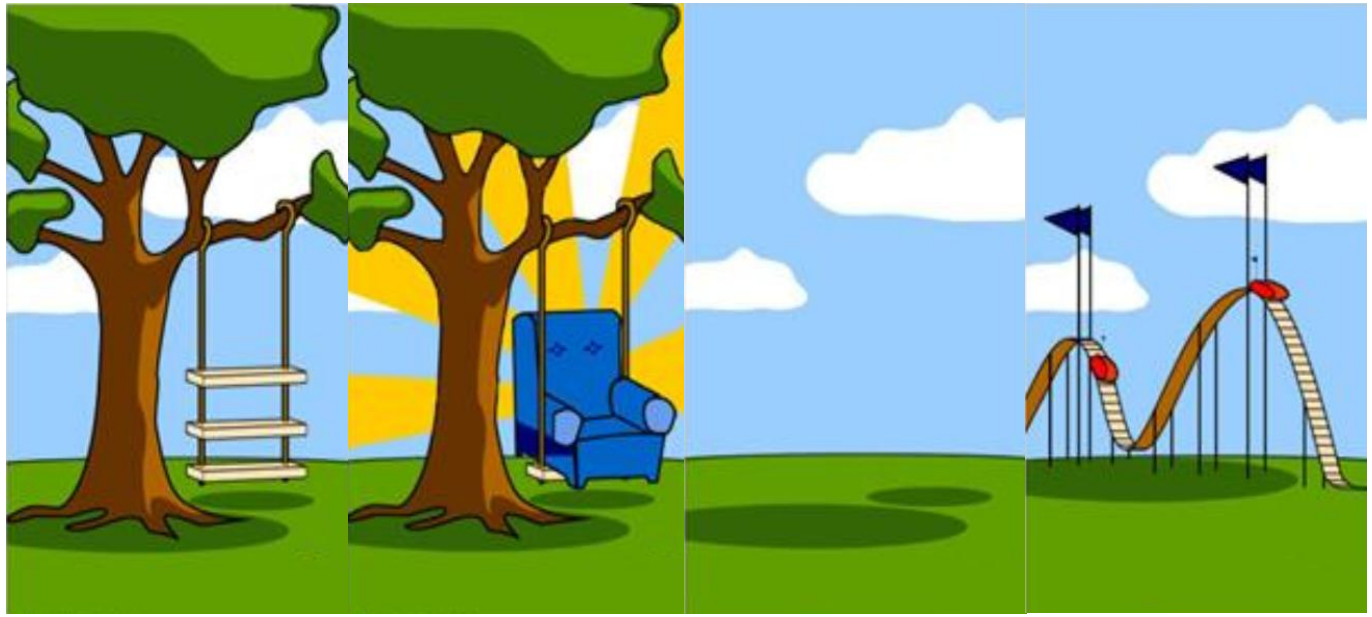
https://www.zentao.pm/blog/tree-swing-project-management-tire-analogy-426.html

Tree Swing spec & after

How the customer explained it	How the business consultant described it	How the project was documented	How the customer was billed	What the customer really needed

https://www.zentao.pm/blog/tree-swing-project-management-tire-analogy-426.html

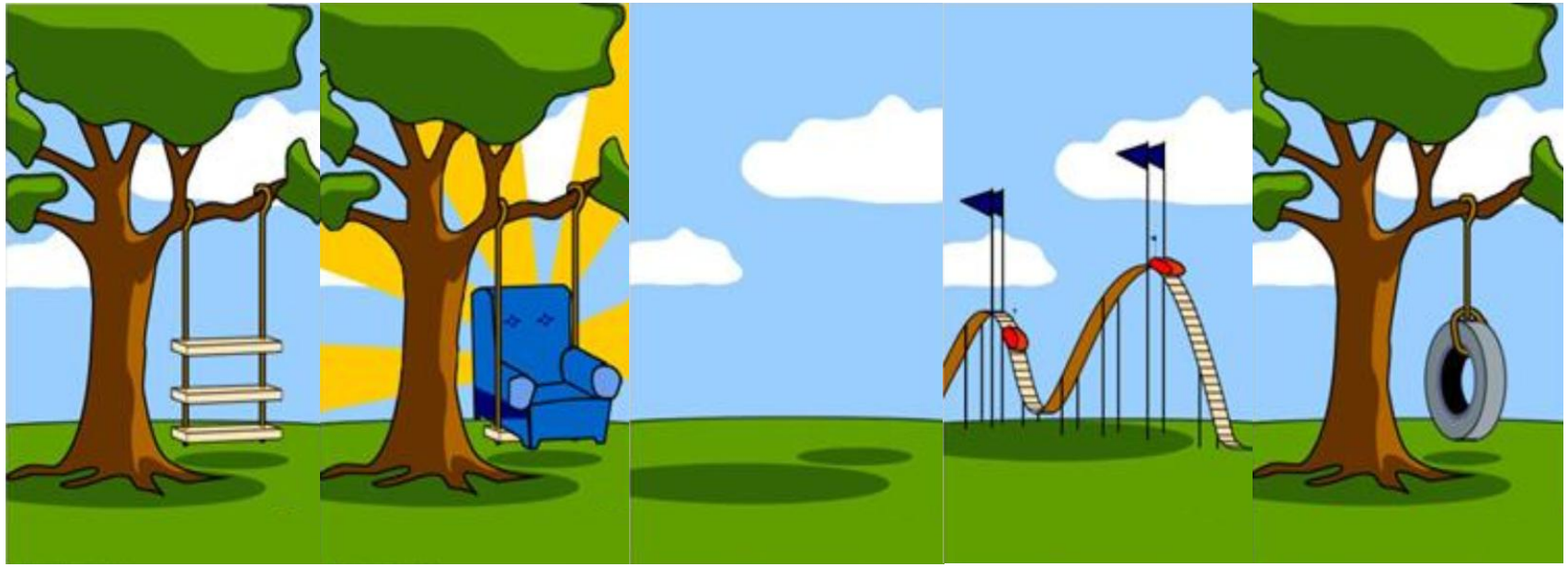
Tree Swing spec & after



How the customer explained it	How the business consultant described it	How the project was documented	How the customer was billed	What the customer really needed

https://www.zentao.pm/blog/tree-swing-project-management-tire-analogy-426.html

Tree Swing spec & after



How the customer explained it	How the business consultant described it	How the project was documented	How the customer was billed	What the customer really needed

https://www.zentao.pm/blog/tree-swing-project-management-tire-analogy-426.html

Analýza, špecifikácia požiadaviek

Aky je problem?

Zlé požiadavky - zlé odhady - zlý plan - predraženy softvér, ktorý robí niečo iné ako chceme

Komunikácia (nie výlučne medzi zákazníkom a vývojárom), ale vo všeobecnosti je často problematickou súčasťou fázy špecifikácie požiadaviek, ale presne nie len

Problém komunikácie požiadaviek

Dokážeme o softvéri komunikovať?

Určite áno, aspoň nejako...

Aj na komunikáciu nám treba štandardy

Ono to aj funguje...

Join at

slido.com

#3647 788

Z ktorej strednej školy prichádzate?

Elektro

0%

Strojn.

0%

Stavebná

0%

Gymnázium

0%

iné

0%

Ako sú na tom iní?

Stavební inžinieri a architekti zvykli formálne vyjadrovať svoje idey, nápady resp. požiadavky vo forme stavebných výkresov, štúdií alebo modelov skutočných budov a podobne, než začali platiť za stavebné práce.

Strojári inžinieri, najprv vytvoria model nového motora, všetky zainteresované strany sa dohodnú na riešení, ktoré podpíšu, a až potom začnú s hromadnou výrobou motora.

Ako sú na tom iní?

V elektrotechnike sa používa formalizmus, ktorý umožňuje čítať modely obvodov a pripojení bez ohľadu na materinský jazyk autora.

Aj v softvérovom inžinierstve máme najprv zachytenú špecifikáciu požiadaviek, ktorá vyzrieva do modelu návrhu v jazyku UML. Až keď sa všetci zainteresovaní dohodnú na návrhu modelu, začnú sa rozsiahle investície do vývoja.

Mnohé softvérové riešenia v uvedených oblastiach umožňujú overovať modely a simulovať ich prevádzku na účely validácie.

Ako sme na tom v oblasti vzdelávania?

Na PSI sa budeme učiť: [zhrnutie]

O vývoji softvéru v celej šírke.

Ako sa vysporiadať s veľkým a zložitým softvérom.

Ako zisťovať a formulovať požiadavky na softvér.

Modelovací jazyk na komunikáciu o softvéri (UML).

Celé to prepojíme vo fimále s implementáciou vybranej časti funkcionality.

Otázky kým pôjdeme ďalej?

Aktivizujte sa!

Pokojne nás prerušujte otázkami.

Vieme, že budete veľa krát zmätení.

Tento predmet je už raz taký: nevieme sa tu toľko oprieť o vaše predchádzajúce skúsenosti, ale pokúsime sa využiť to maximum čo môžme.

Pýtajte sa najmä ak...

- nebude jasné, prečo sa nejakú vec učíte
- sa vám bude zdať, že si v niečom odporujeme
- píšte si poznámky, najlepšie rukou

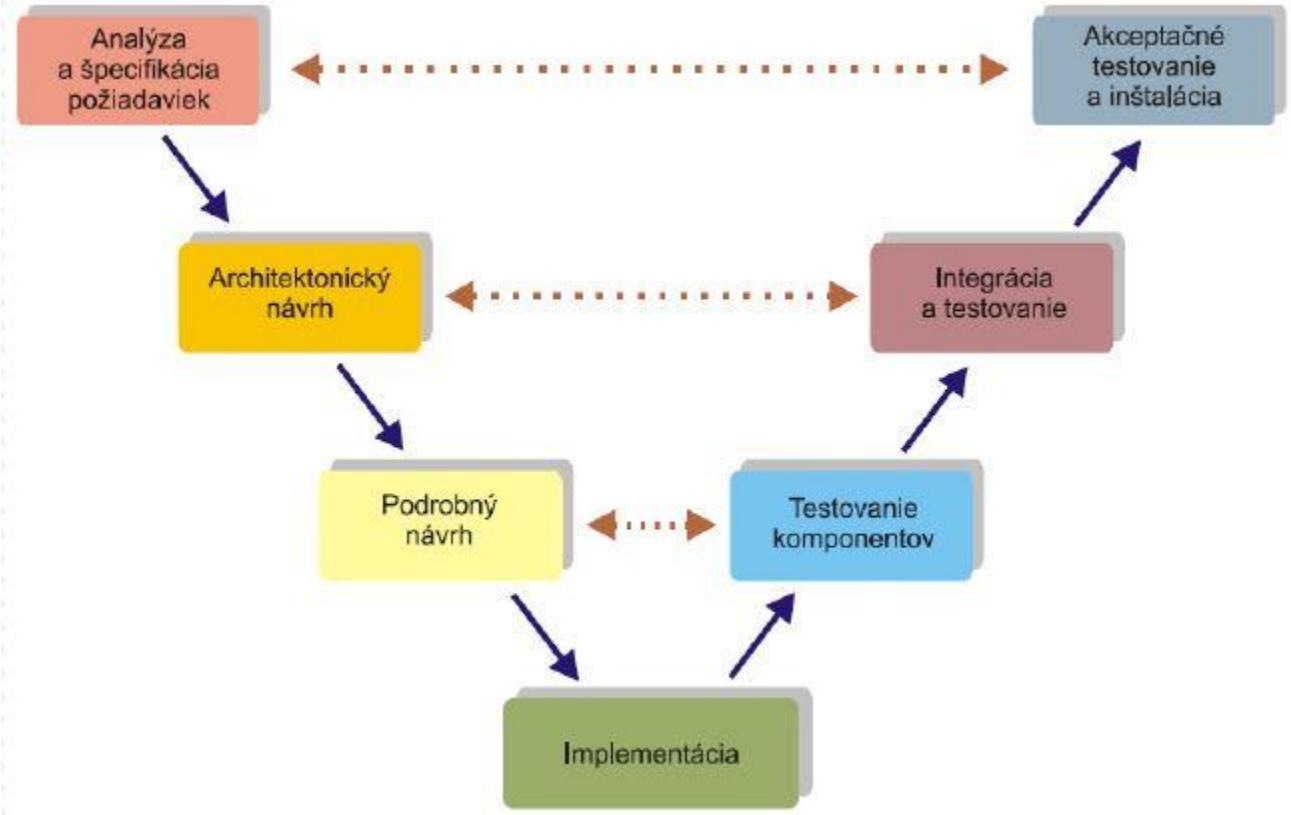
BTW „Repetitio mater studiorum“

Opakovane si prečítajte stránku predmetu

Využívajte možnosť ctrl+f

https://www2.fiit.stuba.sk/~lang/PSI/

Životný cyklus softvéru:
Analýza, špecifikácia požiadaviek (V model)



Životný cyklus softvéru:
Analýza, špecifikácia požiadaviek (V model)

Biznis analýza: zisťovanie potrieb zákazníkov a podmienok v ktorých má softvér fungovať

Špecifikácia požiadaviek: formulovanie toho, čo má softvér robiť a spĺňať

Analýza požiadaviek: zameraná najmä na realizovateľnosť požiadavky

Uvedené činnosti sú neoddeliteľnou súčasťou tvorby akéhokoľvek väčšieho softvéru...

... preto chceme, aby ste si ich techniky vyskúšali

Artefakty vo vývoji softvéru

Ciele

Čo a prečo:

Zákazník definuje svoje potreby sformulovaním biznis cieľov. Bez cieľov projekt nemá zmysel. Všetky artefakty a činnosti v softvérovom projekte by mali súvisieť s ich plnením.

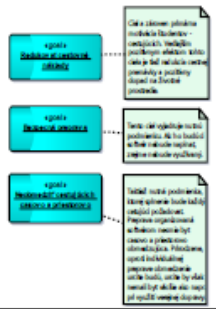
Využívané techniky:

Diagram biznis procesov (non-UML)
Texty

Príklad prvotného zadania:

Pravidelné cestovanie je nákladné. Jednou z možností ako náklady redukovať je spoločná automobilová preprava (car-pooling). Tá však efektívne funguje iba vtedy, keď ju organizuje veľká komunita. V rámci komunity musí zároveň existovať základná miera dôvery. Univerzita toto identifikovala ako príležitosť a chce pre svojich študentov zaviesť systém zdieľania kapacít dní pri cestovaní. Ten má umožniť študentom sa efektívne organizovať v spoločnom cestovaní a zdieľaní nákladov.

Príklad cieľov takéhoto softvéru:



Zainteresované osoby (aktéri)

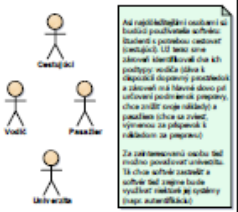
Čo a prečo:

Od prvej chvíle spolu s cieľmi charakterizujeme osoby, ktoré majú na softvéri nejaký záujem (aktérov). Neskôr v analýze sa nezaobídeme bez komunikácie s nimi. Mnohé z nich budú tiež používateľmi vytváraného softvéru.

Využívané techniky:

Diagram biznis procesov (non-UML)
Texty

Príklad identifikovaných aktérov:



Biznis procesy

Čo a prečo:

Postupy, ktoré majú napĺňať stanovené biznis ciele sa nazývajú biznis procesy. Vystupujú v nich tiež aktéri a rôzne ďalšie artefakty. Definovať biznis procesy potrebujeme, aby sme následne mohli identifikovať rolu budúceho softvéru v rámci nich.

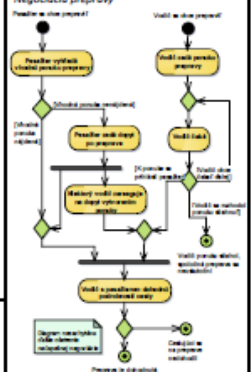
Využívané techniky:

Diagram činností (activity diagram)
Diagram biznis procesov (vzťahy medzi BP)
Texty (dôležité pre opisy diagramov)

Príklady biznis procesov:

Príklad biznis procesu (diagram aktivít):

Negociácia prepravy



Pasažier vyhľadá v existujúcich ponukách prepravy (zadaných predtým vodičmi) najväčšiu, ktorá mu vyhovuje. Môže tiež zadať vlastný dopyt po preprave (želanú cestu), na ktorú môžu vodiči tiež reagovať. Po nájdení potenciálnych spolucestujúcich je ďalšie vyjednanie podrobností ponechané na nich.

Slovník pojmov

Čo a prečo:

Ako vývojári potrebujeme porozumieť doméne, pre ktorú softvér vyvíjame. Súbežne s biznis modelovaním preto vytvárame slovník pojmov. Definované pojmy sa neskôr často ukážu ako funkcie a dátové entity softvéru. Nevyhýbame sa ani „triviálnym" pojmom. Aj tie môžu rôzni ľudia chápať odlišne, čo môže neskôr spôsobiť problémy.

Využívané techniky:

Zoznam textových definícií

Príklad:

- *Negociácia – Vyjednávanie o podmienkach prepravy.*
- *Trasa – opakovane používaná geografická trajektória z miesta A do miesta B*

Funkcionálne požiadavky

Čo a prečo:

Zmapuje sa budúca funkcionalita softvéru. Snaha je reprezentovať ju najmenšími možnými jednotkami, ktoré ešte majú biznis zmysel. Týmito jednotkami sú prípady použitia (use cases) alebo ich obdoba, napr. odfahčené používateľské príbehy (user stories).

Využívané techniky:

Diagram prípadov použitia (use case diagram)
Textové používateľské príbehy

Príklad diagramu prípadov použitia:

Prípady použitia procesu negociácie prepravy

Príklady dvoch používateľských príbehov:

Forma kto-čo-prečo:
Kto: Vodič.
Čo chce spraviť: Vytvoriť ponuku na prepravu, zadaním časov a miest odchodu a príchodu.
Prečo: Aby mohli pasažieri ponuku vyhľadať a prípadne si ju aj vybrať (a znížiť mi náklady).

Alternatívne aj forma ako-chcem-aby:
Ako pasažier, chcem v existujúcich ponukách prepravy vyhľadať pre mňa vyhovujúcu a prihlásiť sa na ňu, aby som sa mohol s vodičom dohodnúť na podmienkach.

Nefunkcionálne požiadavky

Čo a prečo:

Definujú sa požadované vlastnosti softvéru, ktoré nemajú charakter funkcií softvéru. Mali by byť merateľné.

Využívané techniky:

Textový opis (kritérií)

Príklad:

- *Softvér je prístupný len pre študentov a zamestnancov univerzity*
- *Odozva pri všetkých akciách používateľov softvéru je maximálne 0.5 sekúnd*
- *Aplikácia má responzívny dizajn a funguje aj pre smartfóny a tablety.*
- *Aplikácia musí podporovať rozlíšenia od 720 x 1280 pixelov na zariadeniach s uhlopriečkou od 4.8 palcov vyššie.*

Scenáre

Čo a prečo:

Prípady použitia sa detailne opíšu. Najdôležitejšie sú scenáre, určujúce čo presne sa má v prípadoch použitia diať.

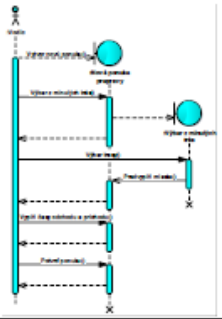
Využívané techniky:

Štruktúrovaný text, číslované zoznamy.
Sekvenčný diagram, diagram činností

Príklad opisu prípadu použitia:

Vytvorenie ponuky na prepravu:
Aktér: Vodič.
Krátky opis: Vodič vytvorí ponuku na prepravu (...).
Vstupné podmienky: vodič je prihlásený
Výstupné podmienky: ponuka je zverejnená
Hlavný scenár:
1. V hlavnom menu vodič vyberie možnosť „vytvor"
2. Otvorí sa obrazovka „vytvorte novú cestu"
3. Vodič zadá miesto začiatku trasy (auto-complete)
4. Vodič zadá cieľové miesto trasy (auto-complete)
5. Vodič zadá čas odchodu
6. Aplikácia zobrazí odhadovaný čas príchodu
7. Vodič stlačí „vytvoriť"
8. Aplikácia zobrazí potvrdenie zverejnenia ponuky
Alternatívny scenár 1:
Po kroku 2 sa vodič rozhodne použiť existujúcu trasu
1.1 Vodič klikne na „výber z minulých tráz"
1.2 Aplikácia zobrazí zoznam minulých tráz
1.3 Vodič zo zoznamu vyberie niektorú z tráz
1.4 Prípad použitia pokračuje bodom 5 hl. toku

Príklad: scenár sekvenčným diagramom



Akceptačné testy

Čo a prečo:

Vytvoria sa testovacie scenáre. Ak nimi softvér prejde, je považovaný za dokončený: sú de-facto zmluvou medzi vývojárom a zákazníkom.

Využívané techniky:

Štruktúrovaný text, číslované zoznamy

Príklad akceptačného testu:

Vytvorenie ponuky na prepravu (Test 01):
Predpoklady testu: Prihlásený používateľ Jožo
Scenár testu:
1. V hlavnom menu vyberte „vytvor"
2. Očakávaná odozva: otvorí sa obrazovka „nová ponuka prepravy"
3. Ako začiatok trasy zadajte „Žilina"
3. Očakávaná odozva: auto-complete po prvom znaku
4. Ako cieľ trasy zadajte „Bratislava"
3. Očakávaná odozva: auto-complete po prvom znaku
5. Ako čas odchodu zadajte „5.3.2016 16:00"
6. Ako čas príchodu zadajte „5.3.2016 18:30"
7. Potvrďte vytvorenie ponuky
Očakávaný stav na konci testu:
- *V databáze je vytvorená nová ponuka*
- *Nová ponuka je vyhľadateľná v rámci scenára „Vyhľadanie ponuky na prepravu"*

Logický model údajov

Čo a prečo:

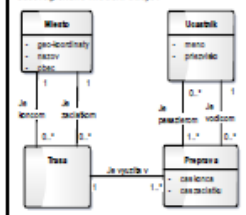
Identifikujú sa údaje, s ktorými softvér pracuje (entity, atribúty, vzťahy, stavy). Nevyhnutný krok pre správne a udržateľné uchovávanie údajov.

Využívané techniky:

Diagram tried
Stavový diagram

Príklady:

Časť logického modelu údajov



Stavový priestor entity Preprava

Obrazovky

Čo a prečo:

Návrh obrazoviek (predloha pre budúci vývoj). Informačná architektúra (prepojenia obrazoviek).

Využívané techniky:

Náčrty obrazoviek (wireframes)
Orientované grafy (informačná arch.)

Príklad:

Návrh obrazovky vytvárania novej prepravy

Architektúra

Čo a prečo:

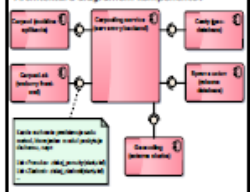
Štruktúrnym pohľadom na softvér „zhora" je jeho architektúra tvorená komponentami, ich opismi, rozhraniami a vzťahmi. Potrebujeme poznať úlohy jednotlivých komponentov. Potrebujeme tiež ukotviť softvér „v priestore", robiť rozhodnutia o výbere technológií, ale aj rozdeľovať prácu do menších celkov či rozdeľovať kompetencie vývojárov.

Využívané techniky:

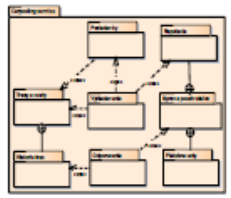
Diagram komponentov
Diagram balíkov
Textové opisy

Príklady:

Architektúra diagramom komponentov



Vnútorná štruktúra súčiastky diagramom balíkov



Integračné testy

Čo a prečo:

Vytvorenie architektúry a rozhraní komponentov softvéru umožňuje napísať prvé automatické testy (integračné). Umožnia priebežne testovať súhru jednotlivých súčastí softvéru.

Využívané techniky:

Programovanie

Príklad:

Test metódy REST rozhrania komponentu CarpoolingService

```
public void
testAnyOpeningPassinator()
  throws ClientException,Exception, IOException{
  // Given
  HttpClientRequest request =
    new HttpGet("http://api.carp.sk/users/jozko");

  // When
  HttpResponse response =
    HttpClientBuilder.create().build()
    .execute(request);

  // Then
  User resource =
    ResourceUtil.serializeResourceFromResponse(
    response, User.class);
  ASSERTTRUE("jozko",
    Matchers.is(resource.getLogin()));
}
```

Rozhrania a kompozícia tried

Čo a prečo:

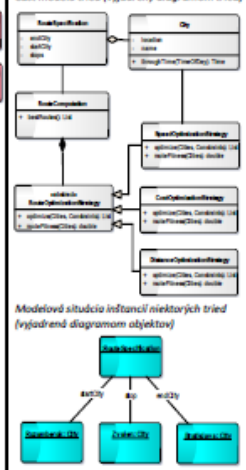
Poznajúc požadovanú funkcionalitu a architektonickú štruktúru softvéru, identifikujú sa programové triedy softvéru, navrhnú sa ich vzťahy a rozhrania (atribúty a deklarácie verejných metód). Niektoré vzniknú ako analógia dátových entít, iné na realizáciu používateľských rozhraní, ďalšie budú držať algoritmy. Identifikáciu tried potrebujeme pre ďalšie dekomponovanie softvéru.

Využívané techniky:

Diagram tried
Diagram objektov
Programovanie

Príklad:

Časť modelu tried (vyjadrený diagramom tried)



Modelová situácia inštancií niektorých tried (vyjadrená diagramom objektov)

Jednotkové testy

Čo a prečo:

Analogicky ako pri integračných testoch, k vytvoreným rozhraniam tried vytvárame jednotkové testy (unit tests). Umožnia sledovať vznik chýb v jednotlivých metódach.

Využívané techniky:

Programovanie

Príklad:

Jednotkový test triedy City, metódy throughTime

```
public class CityTest {
    private City city;
    private Time Building = new Time(0,0,0);

    @Before
    public void setUp() {
        city.diameter = 1;
        city.population = 200000;
        city.main_road_type = Road.highway;
    }

    @Test
    public void testThroughTimeNoLimit() {
        ASSERTTRUE(Math.ignoreTime(throughTime) > 0);
    }
}
```

Implementácia (nielen tried)

Čo a prečo:

Implementáciou triedy rozumieme programovanie samotného „vnútra" triedy. To zahŕňa predovšetkým programovanie vystavených metód, definície pomocných atribútov a metód a pod. V širšom zmysle Implementácia znamená aj ďalšie aktivity spojené s realizáciou softvéru ako napr. programovanie používateľských rozhraní, konfiguračných súborov a pod. Čo všetko? To veľmi záleží od použitých technológií, ale aj typu vytváraného softvéru. Implementáciou sa softvér stáva konkrétnou, použiteľnou vecou.

Využívané techniky:

Programovanie

Fyzický model údajov

Čo a prečo:

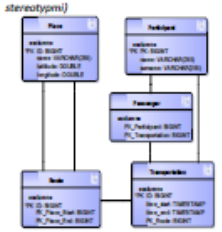
Vytvorí sa fyzický model údajov, ktorým už presne určíme, ako budú v databáze uchovávané údaje. Databázu tak môžeme fyzicky vytvoriť a používať.

Využívané techniky:

Diagram tried
Programovanie

Príklad:

Fyzický model údajov (diagram tried so stereotypmi)



Štruktúra pri nasadení

Čo a prečo:

Uskutočnia sa rozhodnutia o konkrétnom umiestnení (hardvéri, virtuálnom prostredí) nasadeného softvéru, aby mohol byť reálne spustený a používaný.

Využívané techniky:

Diagram rozmiestnenia (deployment diagram)

Príklad:

Rozmiestnenie softvérového systému (diagram rozmiestnenia)

Analýza | Špecifikácia | Návrh | Implementácia

Životný cyklus softvéru:
Analýza, špecifikácia požiadaviek

Toto ste už vlastne začali tento týždeň...

Rrozdelení do skupín (po štyroch), príprava nástrojov EA, Editor textu (PDF)

Konzultácia zámeru vášho projektu. Zvyšok času: rozhovory so „zákazníkom“

Sledovaný cieľ: identifikujte a rozpíšte viacero používateľských príbehov (user story) a jeden prípad použitia, ktorý pokladáte za charakteristický pre váš projekt

Životný cyklus softvéru:
Analýza, špecifikácia požiadaviek

Činnosť, v ktorej analyzujeme určitú oblasť biznisu alebo organizáciu

- Čo sú ciele (biznisu/organizácie)
- Aké činnosti sa pre ich dosahovanie vykonávajú
- Akí ľudia sa na tom podieľajú
- Aké veci sa pri tom používajú

Identifikácia problémov a potrieb v danej oblasti

Hľadanie možných (softvérových) riešení

Biznis analýza…

Nie je návrh softvéru

Nie je dokonca ani špecifikáciou softvéru

Nesmie obsahovať (ani stopu) softvérovo-inžinierskych pojmov

S budúcim softvérom má spoločné len toľko, že v nej identifikujeme, akým cieľom má softvér slúžiť

Prečo robíme biznis analýzu?

Musíme zistiť, čo náš zákazník naozaj potrebuje

... a to môže byť dosť odlišné od toho čo nám povie

Musíme sa porozumieť, ako jeho biznis funguje

... nechceme robiť naslepo (veľmi zlý nápad)

Musíme porozumieť jazyku zákazníka

... a prekonať vzájomnú „komunikačnú bariéru“

Aký bude postup?

Konkretizujete rámcové zadanie = zadefinujete si vlastnú doménu

Budete dávať dohromady ďalšie informácie

- ... informácie o oblasti + požiadavky
- ... najmä rozhovorom s vaším cvičiacim (zákazníkom), vami v dvojroli
- ... štúdiom existujúcich systémov
- ... štúdiom ďalších zdrojov

Tie informácie budete organizovať (modelovať)

- ... najprv neorganizované poznámky
- ... potom diagramy, slovníky, štruktúrované texty ...

Zber informácií

Kto je viac aktívny (Push vs. Pull)?

Zistite, ako veci u zákazníka bežia teraz

Pokúste sa nájsť hranice

- ... biznisu zákazníka (čo ešte rieši, čo už nerieši)
- ... budúcej pôsobnosti vytváraného softvéru

Pokúste sa zistiť motivácie a bolesti zúčastnených

- ... zákazníka
- ... používateľov

Zisťujte priority (nikdy sa nedá spraviť všetko)

Nespoliehajte sa len na rozhovor s cvičiacim

Zber informácií

Najjednoduchší spôsob dokumentovania diskusií - zapisovanie

Neštruktúrovaný text

Použiteľné sú ľubovoľné artefakty

Dôležité je zaznamenať jazykom doménových expertov, jazykom domény problému nie domény riešenia

Žiaden pojem nie je príliš triviálny

(naopak, s pojmami, ktoré sú bežné, je najviac problémov)

Vyhýbajte sa softvérovo-inžinierskym pojmom

Kartotéka - ~~Databáza~~

Čitateľ - ~~Používateľ~~

Neustále dopĺňajte slovník pojmov, Slovník pojmov (glossary)

Používateľské príbehy (User Stories)

Čo sú User Stories?

Jednoduchý opis požiadavky z pohľadu používateľa

Zameranie na hodnotu pre používateľa

Používané najmä v agilnom vývoji

Používateľské príbehy (User Stories)

Štandardná štruktúra/formát: „As a… I want… so that…“

Ako <typ používateľa>
chcem <funkcionalitu>
aby som <získal hodnotu>

Ako registrovaný používateľ
chcem resetovať heslo
aby som získal prístup k svojmu účtu

Používateľské príbehy (User Stories)

Charakteristiky dobrej User Story (INVEST)

- Independent - nezávislá
- Negotiable - diskutovateľná
- Valuable - prináša hodnotu
- Estimable - odhadnuteľná
- Small - primerane malá
- Testable - testovateľná

Používateľské príbehy (User Stories)

Akceptačné kritériá:

Spresňujú podmienky splnenia požiadavky

Slúžia ako základ pre testovanie

Zvyšujú jednoznačnosť interpretácie

Používateľské príbehy (User Stories)

Stručné a orientované na hodnotu

Postupné spresňovanie, Podpora komunikácie v tíme

Rozklad a priorizácia: Epic → User Story → Use Case → Task

Backlog ako zoznam priorít, Iteratívne plánovanie

Lepšia komunikácia so zákazníkom, Flexibilita pri zmene požiadaviek, Podpora iteratívneho vývoja

Obmedzenia a riziká: Riziko straty systémového pohľadu

Životný cyklus softvéru:
Analýza, špecifikácia požiadaviek

Máte vytvoriť elektronický obchod (e-obchod) na základe zoznamu požiadaviek

Aké typické interakcie so systémom - prípady jeho použitia - by sme mohli identifikovať v e-obchode?

Životný cyklus softvéru:
Analýza, špecifikácia požiadaviek

1. Systém musí umožniť zákazníkovi vyhľadať výrobok.
2. Systém musí umožniť zákazníkovi nastaviť počet položiek, ktoré sa naraz zobrazujú. . .
3. Systém musí umožniť zákazníkovi objednať výrobok.
4. Systém musí umožniť zákazníkovi zrušiť objednávku.

...

99. Systém musí umožňovať expedovať objednávku.

...

125. Do systému musí byť možné zadať nové druhy výrobku.

Prípady použitia (Use Cases)

Prípady použitia zachytávajú typické interakcie so systémom.

Ucelené úlohy, ktoré si používateľ kladie za cieľ a ktoré po ich vykonaní spôsobujú zmenu na strane systému alebo používateľa.

Ak bysmemali e-obchod rýchlo sprevádzkovať, ktorý z prípadov použitia by sme mali realizovať ako prvý?

Perspektíva/úroveň/level pohľadu na prípad použitia v kontexte sledovaného cieľa.

Perspektíva/úroveň/level pohľadu na UC

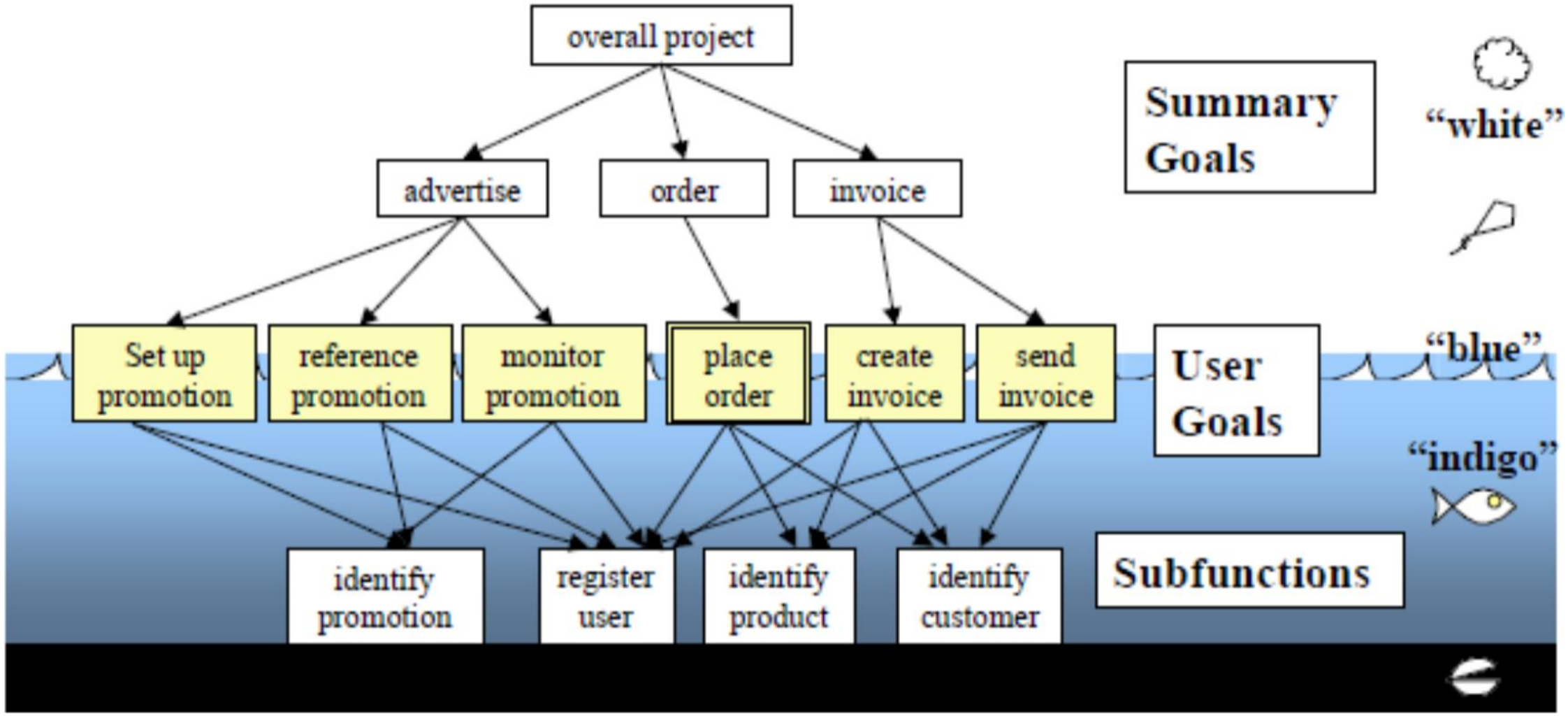
Neodporúčané:

Vysokoúrovňové/sumárne ciele/biznis ciele (napr. zabezpečenie životného cyklu objednávky)

Nízkoúrovňové/pomocné funkcie (napr. zadanie mena a adresy)

Odporúčané:

To, čo vystihuje ciele používateľa, leží medzi vysokoúrovňovými a nízkoúrovňovými (napr. Zadaj objednávku, Vyhľadaj výrobok, Vytvor výrobok, Expeduj objednávku...)



A. Cockburn. Writing Effective Use Cases. Addison–Wesley, 2000.

Prípad použitia: Zadaj objednávku

Zákazník vyberá výrobok do košíka. Ten sa stáva súčasťou jeho objed návky, ktorú nakoniec potvrdí, čím sa objednávka presunie na expedova nie.

1. Zákazník zvolí zadanie objednávky.
2. Systém zobrazí možnosti vyhľadávania.
3. Zákazník nastaví možnosti vyhľadávania a spustí vyhľadávanie.
4. Systém zobrazí vyhľadané položky.
5. Zákazník vyberie z vyhľadaných položiek a potvrdí výber.
6. Systém vloží zvolený výrobok do košíka.
7. Zákazník môže pokračovať vo výbere výrobkov– prípad použitia po kračuje krokom 2.

8. Zákazník objedná výrobky v košíku.

9. Systém vyžiada údaje potrebné na realizáciu objednávky vrátane spô sobu platby.

10. Zákazník zadá požadované platobné údaje.

11. Kedykoľvek počas objednávania, zákazník môže vzdať tento proces.

12. Systém uloží objednávku do zoznamu objednávok na expedovanie.

13. Pre každý výrobok v objednávke, systém zistí stav zásob.

14. Systém uloží záznam do plánu doplnenia zásob o potrebe zvýšenia stavu každého objednaného výrobku, ak by jeho stav po expedovaní po klesol pod stanovený limit.

15. Prípad použitia končí.