Princípy softvérového inžinierstva

Prednáška 3:

Manažment požiadaviek

Mária Bieliková, Jakub Šimko
maria.bielikova@stuba.sk
jakub.simko@stuba.sk

STU FIIT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Dnešný program:

Ukážka EA

Požiadavky

- **Aké majú vlastnosti**
- **Ako ich tvoriť čo najlepšie**

Ukážka vstupných testov

Vlastnosti softvéru (brief)

bit.ly/psi-dotaznik

Na začiatok: ukážka EA

-- ako pridať text (linked document)

-- ako nastaviť generovanie dokumentácie

-- vysvetliť rozdiel medzi modelom a diagramom

bit.ly/psi-dotaznik

Zo spätnej väzby

Ktorú verziu EA si nainštalovať?

najlepšie **13.5,** prežijete aj s 13 aj 14

rozdiely sú v používateľskom rozhraní

Môžeme projekt robiť v angličtine či češtine?

Áno, po dohode s cvičiacim.

v EA bude treba premenovať niektoré balíky

Tatranský čaj? ☺

Len voda. Fľašu používam v rámci Zero Waste.

https://www.youtube.com/watch?v=tb4uCeJkW1o

bit.ly/psi-dotaznik

Zo spätnej väzby

UTF-8 v EA ?

Treba zapnúť slovenčinu pre non-unicode programy (Windows-specific záležitosť)

https://www.digitalcitizen.life/changing-display-language-used-non-unicode-programs?utm_source=7tutorials.com&utm_medium=redirect&utm_campaign=7_Tutorials_Redirect

bit.ly/psi-dotaznik

Zo spätnej väzby

Sli.do aj na prednáške PSI?

Výsledky hlasovania by boli viditeľné?
Bolo by viac vyjadrení a boli by reprezentatívnejšie.

Ad hlasovanie: uznávam (vyskúšam)

Ad viac vyjadrení: razím inú filozofiu

Prečo máme PSI súbežne s DBS?

(PSI by sa hodilo skôr, lebo v DBS už máme využívať poznatky z PSI)
(konkrétne šlo o návrh používateľských scenárov)

Vajce-sliepka (závislosť bude aj opačná)

Tlak na posun PSI je z obidvoch strán

bit.ly/psi-dotaznik

Zo spätnej väzby

„Prvú hodinu a pol ste kecali o veciach ktoré nemajú podla mna so softverovym inzinierstvom nic spolocne,

→ čo je potom softvérové inžinierstvo?

pricom mala byt tema prednasky UML a ako pracovat s enterprise architect, pričom ste to odignorovali a venovali ste sa tomu az na konci priblizne 15 minut.

→ harmonogram sa môže meniť a aj sa zmenil

→ vždy sa snažím dávať do prednášok to, čo je práve najviac potrebné

Keby som chcel pocuvat tie veci ktore odzneli v prvej hodine a pol, tak by som si preniesol MIPko alebo PAMko, pretoze to bolo uplne rovnake a uplne rovnako zbytocne.“

→ tak si počkajte 10 rokov... potom mi to príďte povedať znovu

bit.ly/psi-dotaznik

Čo je to softvérové inžinierstvo?

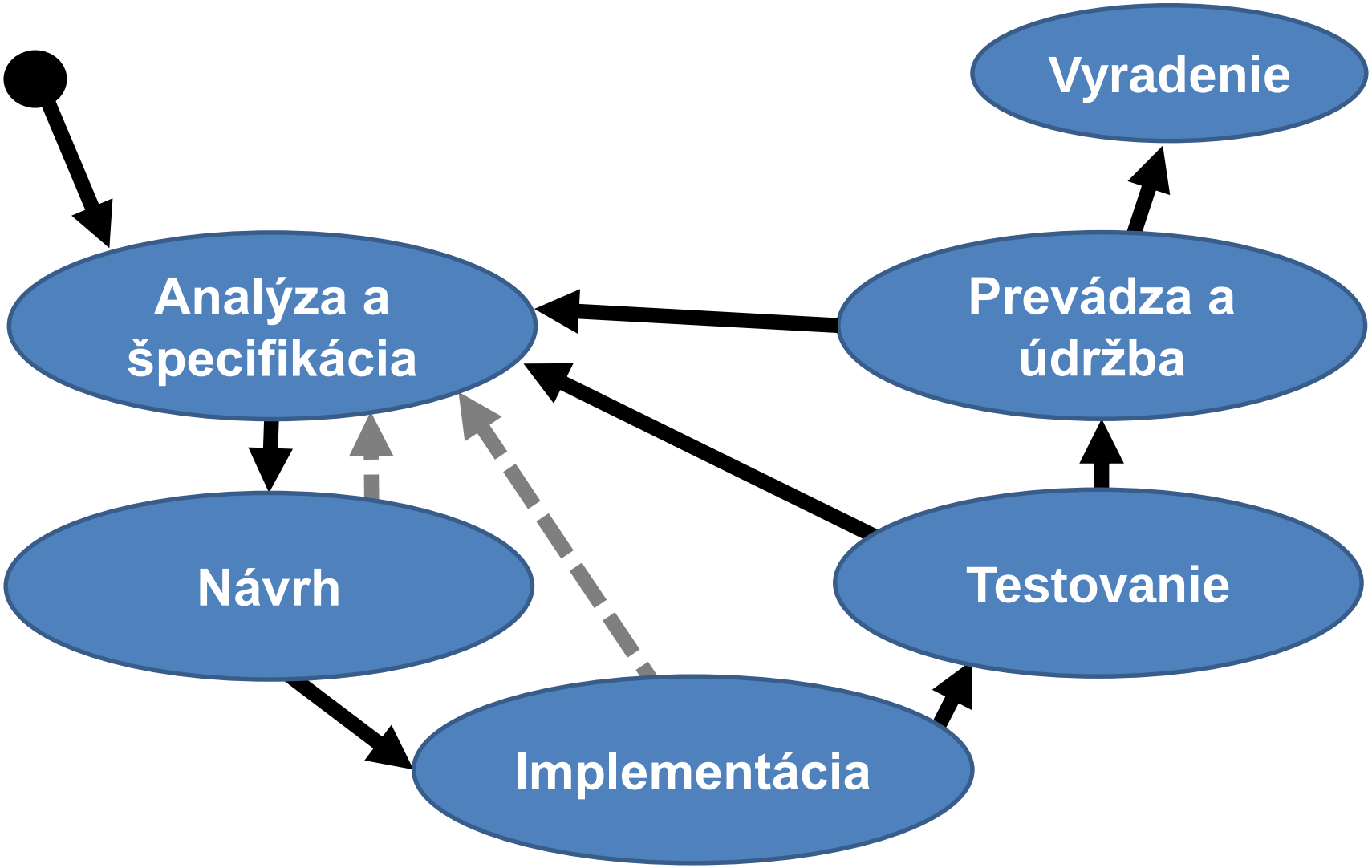
Je to disciplína, systematicky sa zaoberajúca vývojom, údržbou, prevádzkou a vyradením softvéru.

Čo je to softvér?

1. programy
2. postupy ako ich používať
3. dokumentácia
- ***špecifikácia programov (požiadavky)***
- ***scenáre použitia***
- ***testovacie scenáre a údaje***
- ***opisy návrhu (architektúry, komponentov, algoritmov, ...)***
- ***príručky***
- ***...***

bit.ly/psi-dotaznik

Pod softvérové inžinierstvo spadá všetko čo sa robí vo vecnej časti softvérových projektov.



bit.ly/psi-dotaznik

STU FIIT

Aj zber požiadaviek je softvérové inžinierstvo a závisí od neho úspech zvyšku projektu

Faktor zlyhania softvérhového projektu	*Výskyt %*
1. Nedostatok vstupov od používateľov	12.8
2. Nekompletné požiadavky a špecifikácia	12.3
3. Meniace sa požiadavky a špecifikácia	11.8
4. Nedostatočná podpora manažmentu	7.5
5. Technologická nekompetentnosť	7.0
6. Nedostatok zdrojov (predovšetkým ľudí)	6.4
7. Nerealistické očakávania	5.9
8. Nejasné ciele	5.3
9. Nerealistické časové ohraničenia	4.3
10. Nová (neznáma) technológia	3.7
Iné	23.0

https://www.codeproject.com/articles/20488/why-software-projects-tend-to-fail

bit.ly/psi-dotaznik

Aj zber požiadaviek je softvérové inžinierstvo a závisí od neho úspech zvyšku projektu

Faktor zlyhania softvérhového projektu	*Výskyt %*
1. Nedostatok vstupov od používateľov	12.8
2. Nekompletné požiadavky a špecifikácia	12.3
3. Meniace sa požiadavky a špecifikácia	11.8
4. Nedostatočná podpora manažmentu	7.5
5. Technologická nekompetentnosť	7.0
6. Nedostatok zdrojov (predovšetkým ľudí)	6.4
7. Nerealistické očakávania	5.9
8. Nejasné ciele	5.3
9. Nerealistické časové ohraničenia	4.3
10. Nová (neznáma) technológia	3.7
Iné	23.0

https://www.codeproject.com/articles/20488/why-software-projects-tend-to-fail

bit.ly/psi-dotaznik

... a závisí aj od manažmentu

Faktor zlyhania softvérhového projektu	*Výskyt %*
1. Nedostatok vstupov od používateľov	12.8
2. Nekompletné požiadavky a špecifikácia	12.3
3. Meniace sa požiadavky a špecifikácia	11.8
4. Nedostatočná podpora manažmentu	7.5
5. Technologická nekompetentnosť	7.0
6. Nedostatok zdrojov (predovšetkým ľudí)	6.4
7. Nerealistické očakávania	5.9
8. Nejasné ciele	5.3
9. Nerealistické časové ohraničenia	4.3
10. Nová (neznáma) technológia	3.7
Iné	23.0

https://www.codeproject.com/articles/20488/why-software-projects-tend-to-fail

bit.ly/psi-dotaznik

„Můžeme o tom vést spory,
můžeme s tím nesouhlasit,
ale to je tak všechno,
co se proti tomu dá dělat.“

bit.ly/psi-dotaznik

Samozrejme, že treba vedieť

programovať
rozbehať nemožné
navrhnúť systému správnu architektúru
hľadať úzke hrdlá
písať čitateľný kód
lokalizovať chyby
optimalizovať výkon
...
(čokoľvek ďalšie čo je vám sdrcu blízke)

bit.ly/psi-dotaznik

Lenže nič z tohto vás nezachráni, keď neviete poriadne, čo máte dodať.

Začnete veci prerábať ...

... nepôjde to, lebo na to ste ten kód nepripravili ...

... prestanete stíhať, bude vás to frustrovať ...

... neustále zmeny či „prekvapivé zistenia“ budú predlžovať a predražovať celý projekt.

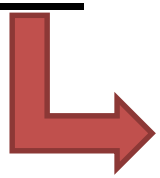
Spýtajte sa ktoréhokoľvek project leada

Manažment požiadaviek (starostlivosť o požiadavky)

bit.ly/psi-dotaznik

Čo je to kvalita softvéru?
Kedy je softvér kvalitný?

Miera naplnenia potrieb zákazníka



prečo nie požiadaviek?

Prečo miera?

Čo myslíte, že sa robí v praxi častejšie?

Validácia, či softvér napĺňa potreby?

alebo

Verifikácia, či softvér spĺňa požiadavky?

Validácia voči potrebám je autentickejšia, ale veľmi náročná.

Verifikácia voči požiadavkám, je oveľa jednoduchšia.

bit.ly/psi-dotaznik

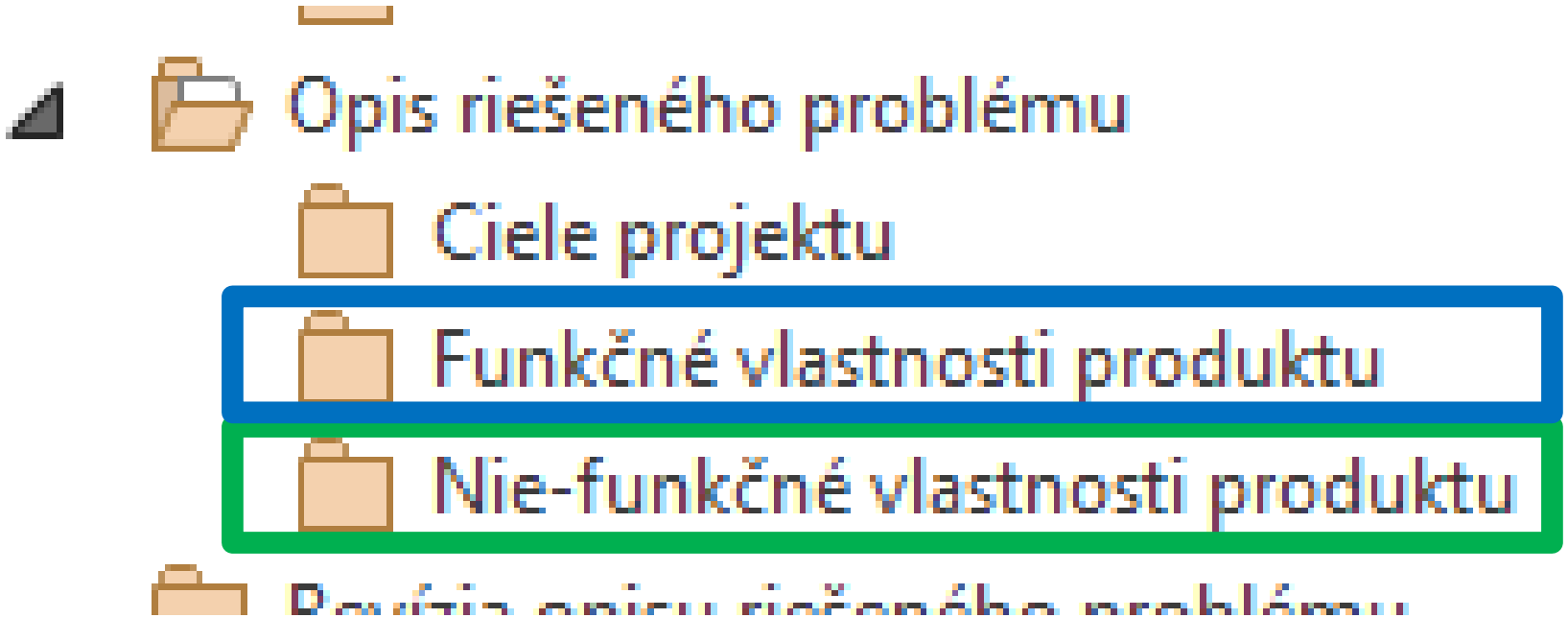
Keď už sme odkázaní na používanie požiadaviek, chceme aby čo najviac odrážali potreby zákazníka

Ukážeme si zopár techník, ako to dosiahnuť...

Pozrieme sa na ne skrz vlastnosti požiadaviek, o ktoré sa treba snažiť

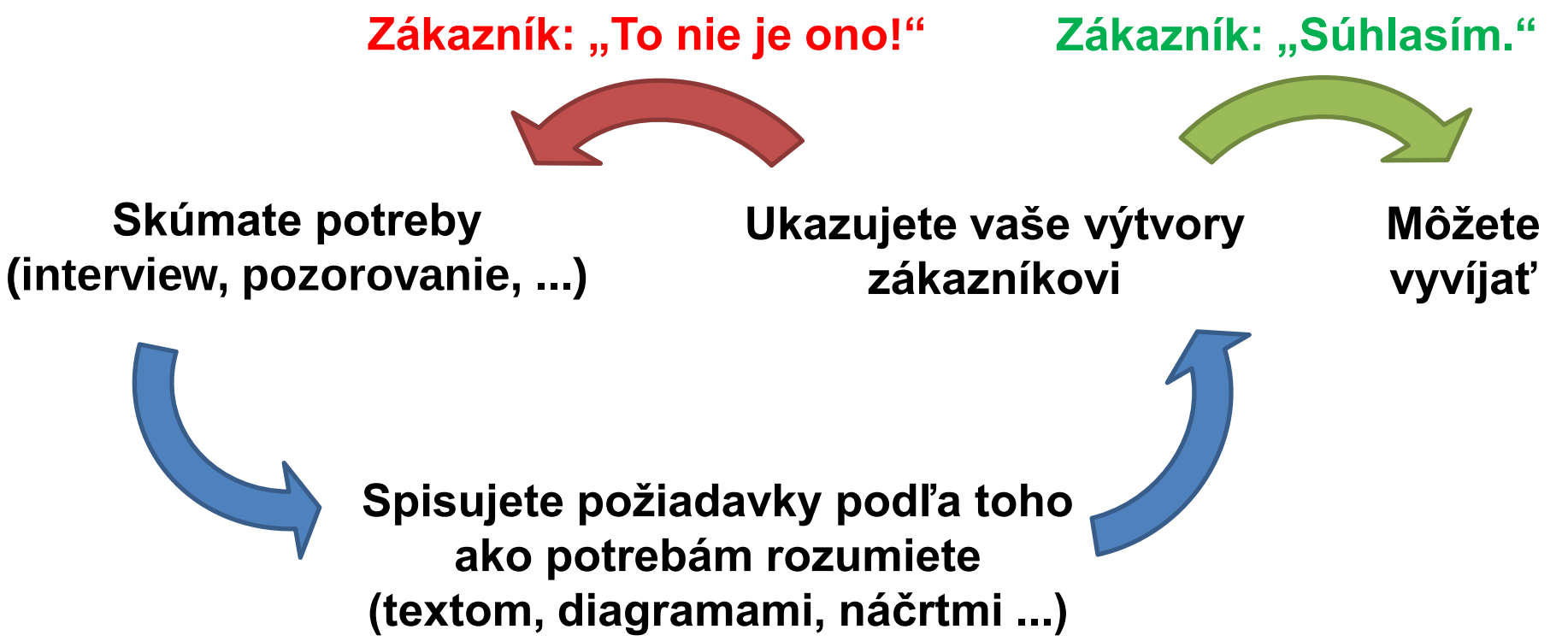
bit.ly/psi-dotaznik

Poznámka: požiadavky rozdeľujeme na funkčné a nie-funkčné



bit.ly/psi-dotaznik

Základ je iterovať: zostavovanie požiadaviek sa vždy deje vo viacerých kolách



Intenzívne iterovanie je veľmi účinné, skoro odhalené chyby je veľmi lacné opravovať.

Iteratívna tvorba požiadaviek je najlepším spôsobom ako zabezpečiť ich správnosť

Správnosť

požiadavka je v súlade s cieľmi a záujmami zákazníka

bit.ly/psi-dotaznik

Čo je nedostatkom týchto požiadaviek?

„softvér musí mať prívetivé používateľské rozhranie“

„softvér musí vydržať nápor väčšieho množstva používateľov“

„softvér musí bežať na viacerých platformách“

bit.ly/psi-dotaznik

Merateľnosť

vieme objektívne vyhodnotiť, že bola požiadavka splnená

bit.ly/psi-dotaznik

Ako zvýšiť merateľnosť týchto požiadaviek?

„softvér musí mať prívetivé používateľské rozhranie“

„nezaškolený používateľ musí úspešne použiť funkciu XY za menej než 5 minút“

„softvér musí vydržať nápor väčšieho množstva používateľov“

„softvér zvládne viac než 1000 požiadaviek zobrazenia používateľského profilu za sekundu“

„softvér musí bežať na viacerých platformách“

„softvér musí bežať na platformách A, B, C a D“

bit.ly/psi-dotaznik

Požiadavky majú čítať a plniť vývojári. Vo vzťahu k nim majú význam dve vlastnosti požiadaviek:

Zrozumiteľnosť

požiadavke vývojári rozumejú
požiadavka nezvádza k nesprávnemu pochopeniu

Jednoznačnosť

požiadavka je sformulovaná tak, že nedovoľuje viaceré interpretácie

bit.ly/psi-dotaznik

Príklady zlej zrozumiteľnosti požiadaviek

„Softvér umožní export VID do RDF“

→ *definovať skratky*

„zobrazí sa index deficitu v poslednom fiškálnom roku“

→ *pojmy by mal vysvetliť slovník (glossary), ktorý by ste mali neustále udržiavať*

„Softvér zabezpečí sprostredkovanie vinkulácie“

→ *vyžaduje veľmi podrobné vysvetlenie pojmu, pravdepodobne na úrovni postupu*

bit.ly/psi-dotaznik

Príklady zlej zrozumiteľnosti požiadaviek

„Softvér umožní export VID do RDF“

Zrozumiteľnosť zabezpečíte, ak budete podrobne modelovať procesy a veci

+ každý vývojár s minimom pudu sebazáchovy vás na nezrozumiteľnosť upozorní

➢ pojmy by mal vysvetliť slovník (glosár)

➢ vyžaduje veľmi podrobné vysvetlenie pojmu, pravdepodobne na úrovni postupu

bit.ly/psi-dotaznik

Na rozdiel od zlej zrozumiteľnosti je viacznačnosť zákernejšia: nedá sa ľahko vidieť

„dorobiť klávesové skratky podľa Chrome“

kontext: projekt prehliadača pre nevidiacich

Študent: *„túto požiadavku v tomto šprinte asi nestihneme“*

Čo tá požiadavka vlastne mala znamenať?

Ja:
„existujúca funkcionalita sa namapuje na skratky“
(no big deal)

Študent:
„všetky skratky podľa Chrome musíme mať implementované“
(robota asi na mesiac)

bit.ly/psi-dotaznik

Jednoznačnosť, podobne ako zrozumiteľnosť, dosahujeme podrobným opisom požiadaviek

1. Ako presne sa má softvér správať navonok
   (ako majú vyzerať scenáre použitia, obrazovky)
2. Čo presne sa ide meniť vo vnútri softvéru
   (ktoré komponenty, triedy, ...)

Ak neviete, či sú požiadavky už dosť presné, vyskúšajte odhadnúť úsilie, ktoré vyžadujú

bit.ly/psi-dotaznik

Planning poker: Každý si pomyslí svoj odhad. Odhady sa naraz zverejnia. A rozprúdia diskusiu.

bit.ly/psi-dotaznik

Ešte je tu jedna vlastnosť, ktorú chceme pri požiadavkách dosahovať:

Sledovateľnosť

v priebehu vývoja vieme požiadavku priradiť vývojovým aktivitám, ktoré smerujú k jej naplneniu

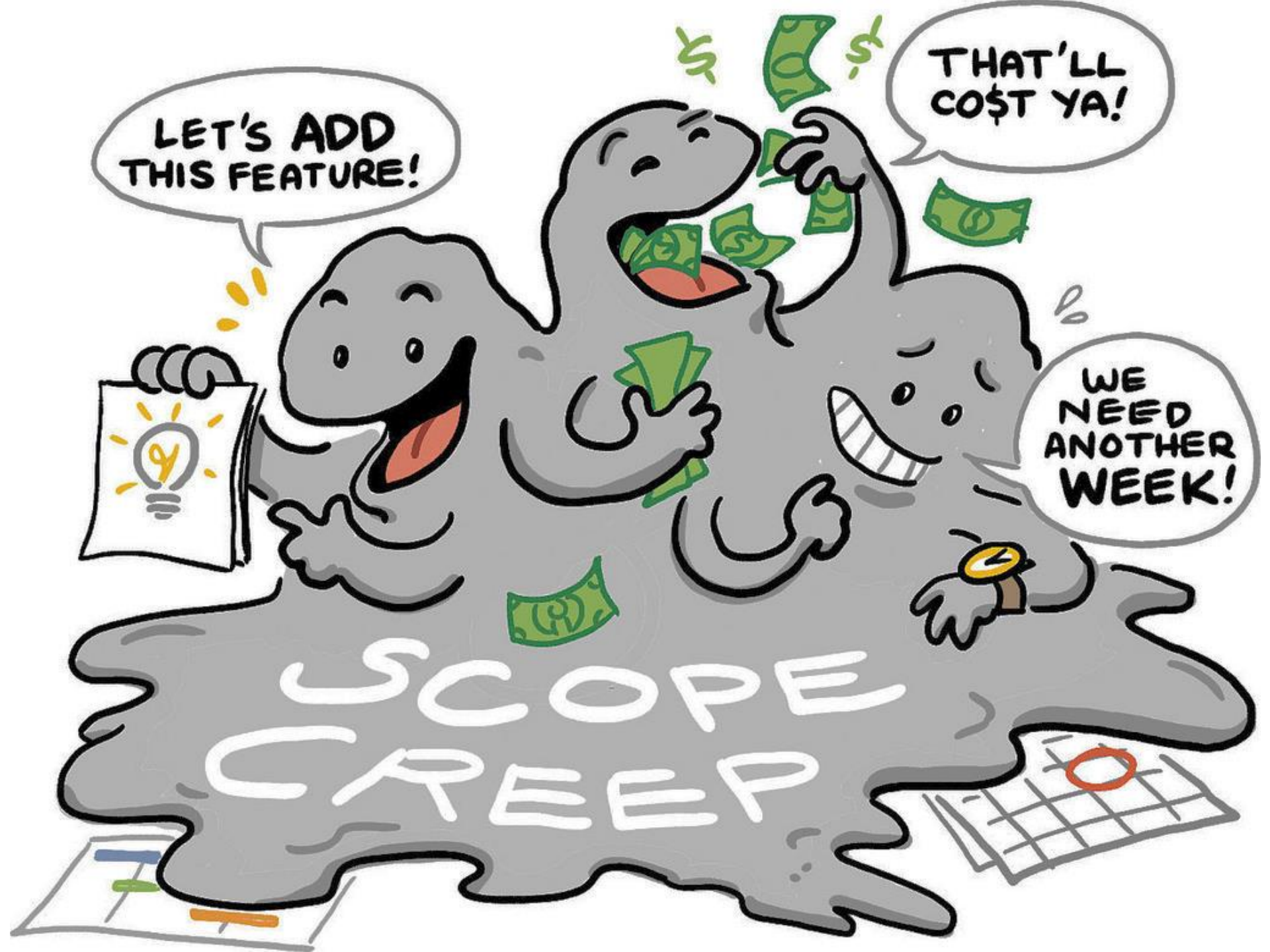
„túto metódu programujem lebo... no vlastne ani neviem“

Sledovateľnosť pomáha strážiť:

→ či robíme to čo máme

→ či nerobíme to čo nemáme

bit.ly/psi-dotaznik


LET'S ADD THIS FEATURE!
THAT'LL CO$T YA!
WE NEED ANOTHER WEEK!
SCOPE CREEP

Vlastnosti
individuálnych požiadaviek

- Správnosť
- Merateľnosť
- Zrozumiteľnosť
- Jednoznačnosť
- Sledovateľnosť

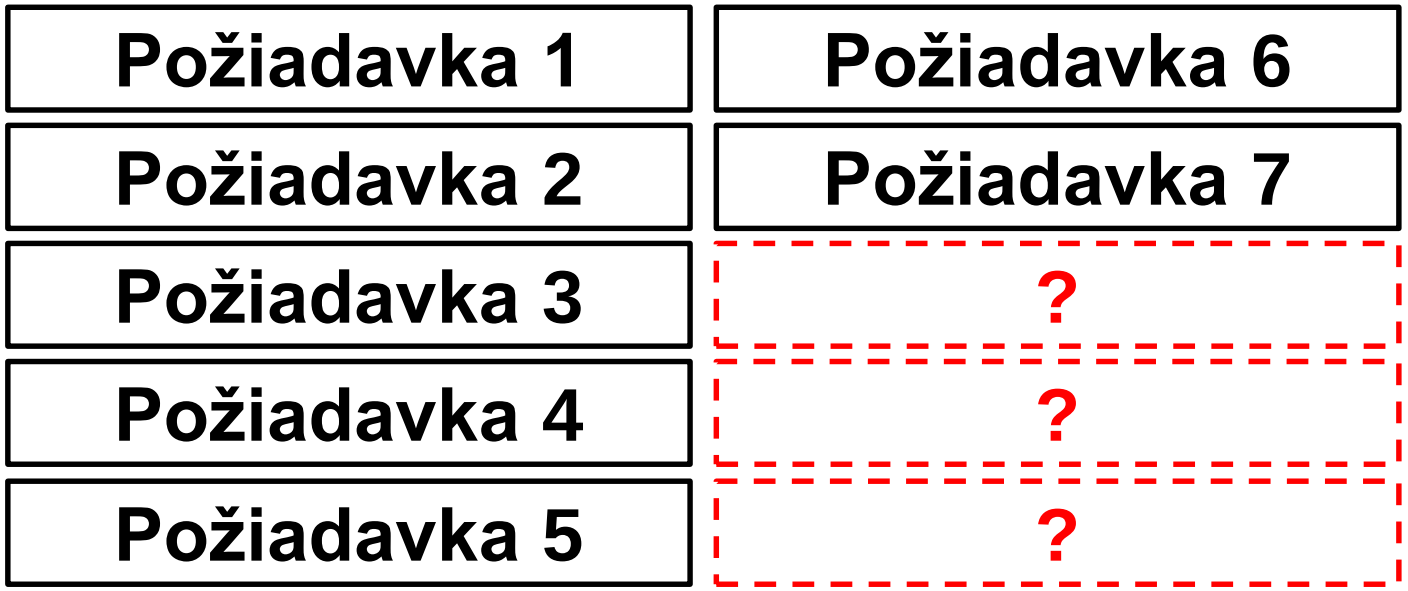
Vlastnosti
celej špecifikácie

- Úplnosť
- Konzistentnosť
- Zoradenie podľa dôležitosti (priority)

bit.ly/psi-dotaznik

Úplnosť

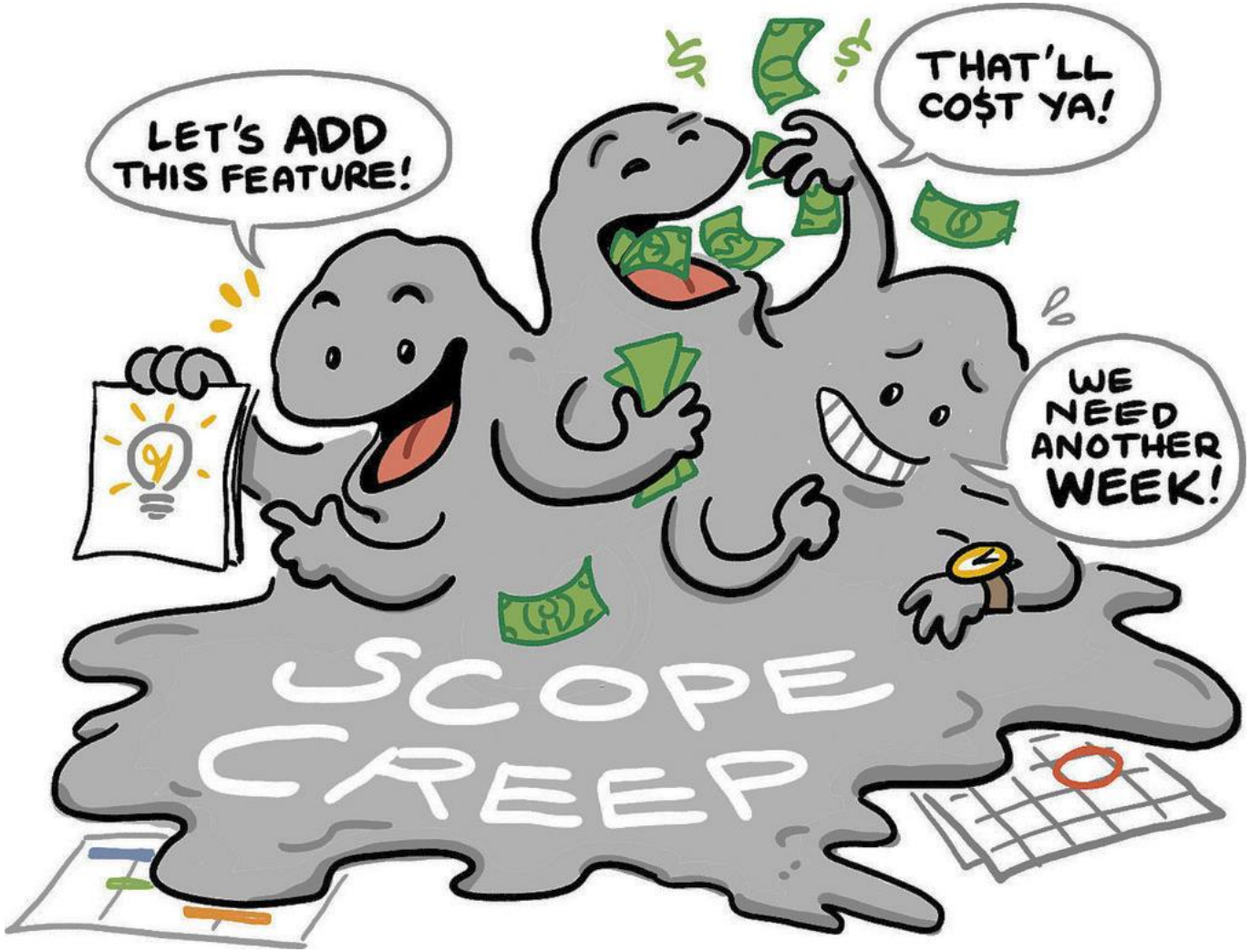
Požiadavky spoločne pokrývajú všetky záujmy a ciele, ktoré chce zákazník vytvorením a používaním softvéru dosiahnuť.



Aké sú dôsledky neúplnosti požiadaviek?

bit.ly/psi-dotaznik

Bez úplnosti nevieme spoľahlivo odhadovať a požiadavky sa začnú ukazovať až počas projektu



bit.ly/psi-dotaznik

Neúplnosť požiadaviek v živote programátorov

Robíte mzdový softvér a programujete výpočet výšky odvodu zamestnanca do Sociálnej poisťovne.

Začiatok projektu: ***výška odvodu závisí od výšky zárobku, plus má nejakú minimálnu úroveň.***

Po mesiaci: ***tie odvody sú vlastne dva, jeden platí zamestnávateľ, druhý zamestnanec.***

Jedna metóda

Po troch mesiacoch: ***dôchodcovia, invalidní dôchodcovia a matky na materskej majú ten vzorec trochu iný...***

Prskajúc to refaktorujete na *strategy*, prípadne *template method*

Po šiestich mesiacoch: ***parlament prijal zákon, ktorý pravidlá mení. Odteraz sa treba pozerať, či zamestnanec nemá aj živnosť. Minimálny odvod už nie spoločný pre všetkých a závisí od troch nezávislých faktorov. Boli pridané ďalšie výnimky...***

Toto už si pýta inú paradigmu zápisu biznis logiky.
Beda každému, kto by toto udržiaval v aplikačnom kóde.

K úplnosti sa môžeme blížiť

Kladením správnych otázok (minulá prednáška)

Poctivou biznis analýzou, pretože viete na čo sa pýtať

- máte vedomosť o biznis procesoch
- viete o existujúcich softvéroch
- budete vedieť, čo je stabilné a čo sa zvykne meniť

Veľmi pomáha prítomnosť skúseností (experta)

Napriek snahe sa úplnosť v praxi nedá dosiahnuť !

bit.ly/psi-dotaznik

Konzistentnosť

Žiadne požiadavky navzájom nie sú v rozpore.

Rozpory existujú na viacerých úrovniach:

Rozpor v biznis logike:

„potvrdenie XY sa vydá na základe preukazu“

„preukaz sa vydá na základe potvrdenia XY“

Viem, smiešne.
Ale fakt sa to stáva.

Rozpor vonkajšom prejave softvéru:

„objednávkový formulár sa musí zmesiť na obrazovku“

„obchodné podmienky sa v plnom znení zobrazujú klientovi v čase, keď ich potvrdzuje.“

„pri objednávke klient potvrdzuje obchodné podmienky“

bit.ly/psi-dotaznik

Konzistentnosť

Žiadne požiadavky navzájom nie sú v rozpore.

Rozpory existujú na viacerých úrovniach:

Rozpor vo vnútri softvéru:

V softvéri si dva komponenty vymieňajú informácie o vzdialenosti. Jeden používa metrické jednotky, druhý imperiálne...

Výsledok: stratili ste sondu za 125 miliónov

Konzistentnosť požiadaviek naberá na dôležitosti pri veľkých projektoch, kde sa práca delí.

bit.ly/psi-dotaznik

Konzistentnosť požiadaviek dosahujeme:

pozeraním sa na veci z viacerých uhlov:

- **cez procesy**
- **cez veci**
- **cez ľudí**
- **cez ciele**

Robíme tak skúšku správnosti

neskôr:

- **cez architektúru**
- **cez dátové modely**
- **cez scenáre**

a potom ešte integračným testovaním

bit.ly/psi-dotaznik

Zoradenie podľa dôležitosti (priority)

Požiadavky sú jednoznačne zoradené a poradie indikuje ich dôležitosť pre zákazníka

80:20

Paretovo pravidlo

Priority sa určujú pomerne jednoducho.
(viď otázky z minulej prednášky)

Treba si ale posedieť so zákazníkom.

bit.ly/psi-dotaznik

Ktoré vlastnosť je najdôležitejšia? Ktorú uprednostniť pred ktorou?

Vlastnosti individuálnych požiadaviek

- Správnosť
- Merateľnosť
- Zrozumiteľnosť
- Jednoznačnosť
- Sledovateľnosť

Vlastnosti celej špecifikácie

- Úplnosť
- Konzistentnosť
- Zoradenie podľa dôležitosti (priority)

bit.ly/psi-dotaznik

Ktoré vlastnosť je najdôležitejšia? Ktorú uprednostniť pred ktorou?

Vlastnosti
individuálnych požiadaviek

- Správnosť
- Merateľnosť
- Zrozumiteľnosť
- Jednoznačnosť
- Sledovateľnosť

Vlastnosti
celej špecifikácie

- Úplnosť
- Konzistentnosť
- Zoradenie podľa dôležitosti (priority)

bit.ly/psi-dotaznik

Vlastnosti softvéru

bit.ly/psi-dotaznik

Existuje veľa vlastností softvéru, pre zákazníka aj vývojárov môžu mať rôznu dôležitosť.

Správnosť **(voči špecifikácii; angl. *correctness*)**
miera splnenia špecifikácie,
najmä *funkcionálnych* požiadaviek.

Spoľahlivosť **(angl. *reliability*)**
frekvencia „nezlyhania“ softvéru (napr. koľkokrát z 1000 načítaní stránky načítanie nezlyhá). Opačnou mierou je *chybovosť* (ako frekvencia „zlyhania“).

Robustnosť **(angl. *robustness*)**
schopnosť softvéru zotaviť sa z chybových stavov, resp. neštandardných vstupov.

bit.ly/psi-dotaznik

Existuje veľa vlastností softvéru, pre zákazníka aj vývojárov môžu mať rôznu dôležitosť.

Efektívnosť **(angl. *efficiency*)**
nenáročnosť prevádzky softvéru v zmysle spotreby strojového času a pamäťového priestoru

Dostupnosť **(angl. *availability*)**
koľko percent času prevádzky je softvér k dispozícii

Škálovateľnosť **(angl. *scalability*)**
ako dobre sa softvér dokáže vysporiadať s nárastom záťaže

bit.ly/psi-dotaznik

Existuje veľa vlastností softvéru, pre zákazníka aj vývojárov môžu mať rôznu dôležitosť.

Použiteľnosť **(angl. *usability*)**
miera jednoduchosti používania softvéru. Ide predovšetkým o vlastnosť používateľského rozhrania

Bezpečnosť voči okoliu **(angl. *safety*)**
miera, do akej miery softvér ohrozuje svojich používateľov. Často sa pletie s *bezpečnosťou*.

Bezpečnosť **(angl. *security*)**
miera, do akej je softvér schopný odolávať útokom zvonku. Často sa pletie s *bezpečnosťou voči okoliu*.

bit.ly/psi-dotaznik

Existuje veľa vlastností softvéru, pre zákazníka aj vývojárov môžu mať rôznu dôležitosť.

Prenosnosť **(angl. *portability*)**
miera, do akej možno softvér používať na rôznych platformách a zariadeniach, resp. miera jednoduchosti takýchto inštalácií.

Interoperabilita **(angl. *interoperability*)**
miera, do akej softvér podporuje spoluprácu s inými softvérmi.

Znovupoužiteľnosť **(angl. *reusability*)**
miera, do akej možno softvér (alebo jeho časti) použiť ako súčasť iného softvéru.

bit.ly/psi-dotaznik

Existuje veľa vlastností softvéru, pre zákazníka aj vývojárov môžu mať rôznu dôležitosť.

Udržovateľnosť **(angl. *maintainability*)**
do akej miery je jednoduché počas prevádzky softvéru pre softvérových údržbárov softvér udržiavať v chode, opravovať chyby.

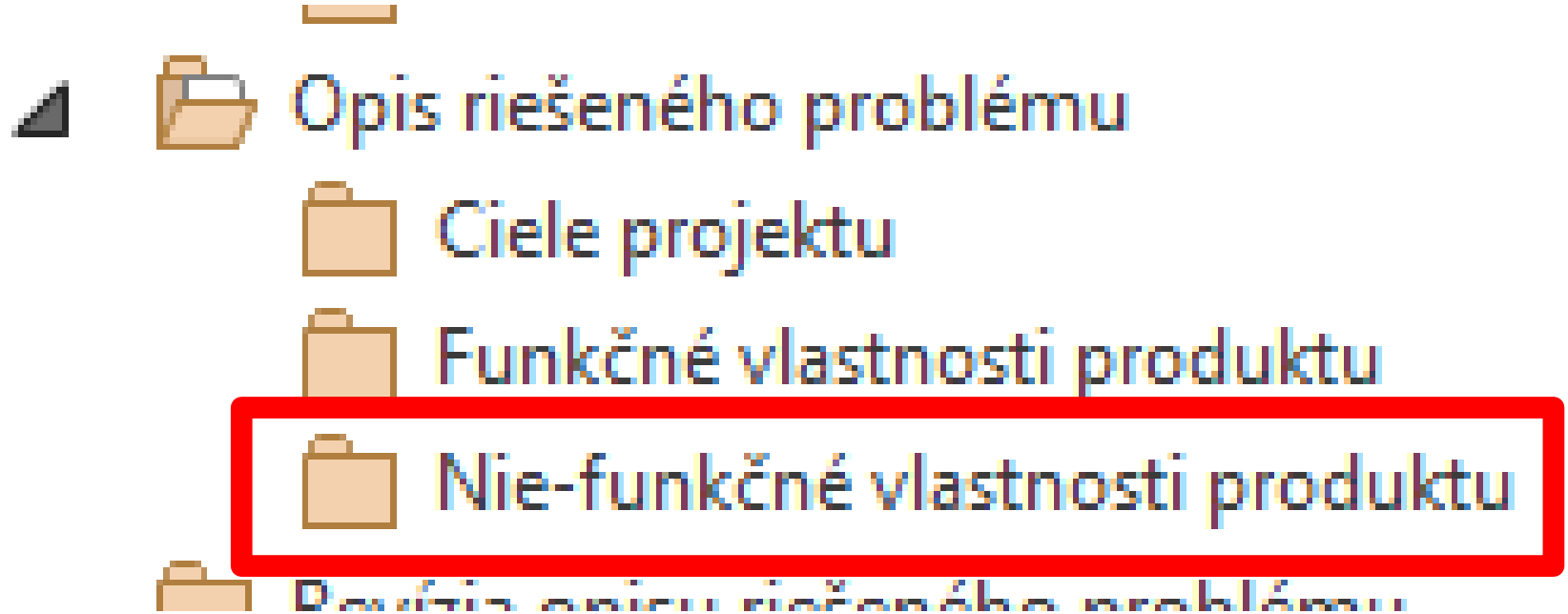
Modifikovateľnosť **(angl. *modifiability*)**
do akej miery jednoducho pridávať novú resp. meniť existujúcu funkcionalitu softvéru.

Testovateľnosť **(angl. *testability*)**
do akej miery možno (efektívne) verifikovať či softvér zodpovedá špecifikácii.

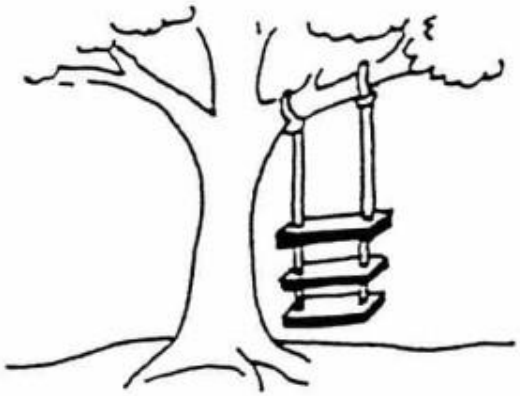
Dokumentovateľnosť **(angl. *documentability*)**
do akej miery možno k softvéru vytvoriť dokumentáciu

bit.ly/psi-dotaznik

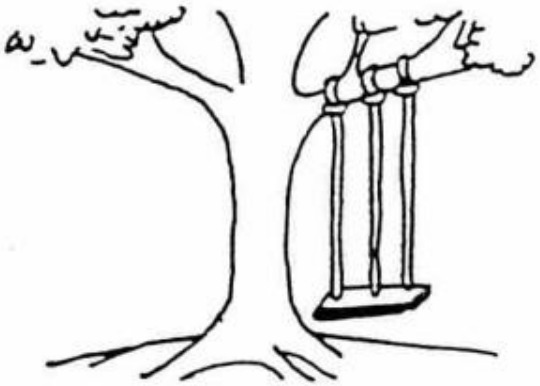
Prejdite si otázku 1.2.13 o vlastnostiach softvéru pri vytváraní časti o nie-funkčných požiadavkách



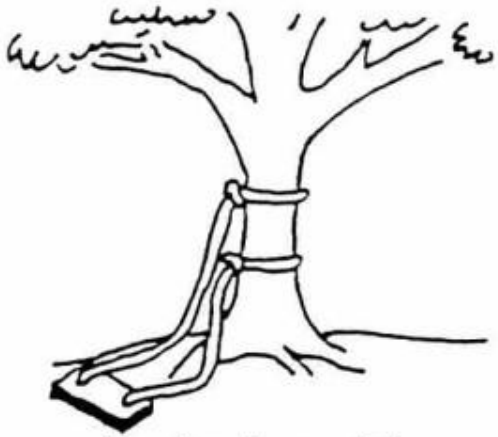
bit.ly/psi-dotaznik



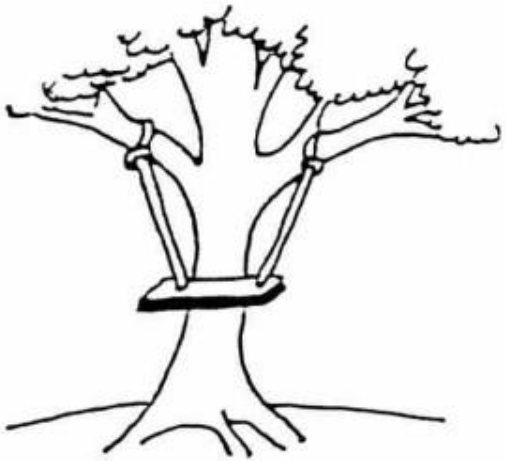
As proposed by the project sponsors



As specified in the project request

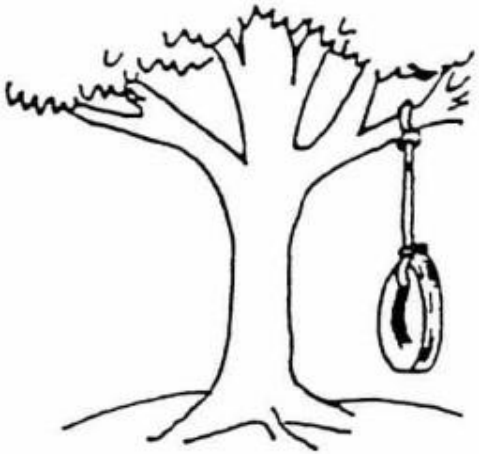


As designed by the senior analyst



As produced by the programmers

As installed at the user's site



What the user wanted

Tree Swing graphic by S Hogh 1993 - from Businessballs.com/treeswing.htm 2013

bit.ly/psi-dotaznik