Úvod do softwarového inženýrství

IUS 2019/2020

1. přednáška

Ing. Radek Kočí, Ph.D.
Ing. Bohuslav Křena, Ph.D.

23. a 27. září 2019

Organizace předmětu – Přednášky

D105 + D0206	E112 + E104 + E105
1BIA + 2BIA + 2BIB	**1BIB** + 2BIA + 2BIB
pondělí 15:00 – 17:50	**pátek 11:00** – 13:50
	
Ing. Radek Kočí, Ph.D.	Ing. Bohuslav Křena, Ph.D.

- V **pondělí 28. října 2019** přednáška odpadne (státní svátek).
- Pravděpodobně v **pátek 1. listopadu 2019** přednáška odpadne také (kompenzace).
- Posledních 10 minut je vyhrazeno pro studijní koutek.
- Konzultace: fóra v IS FIT, e-mail, osobně
- Děkujeme prof. M. Bielikové za poskytnutí původních přednášek.

Organizace předmětu – Cvičení (1/2)

Témata

1. Specifikace požadavků v UML – diagramy případů užití, diagramy aktivit a stavové diagramy (3. a 4. týden výuky)
2. Datové modelování – ER diagramy (5. až 7. týden výuky)
3. Analýza a návrh v UML – diagramy tříd a diagramy objektů, sekvenční diagramy a diagramy komunikace (8. a 9. týden výuky)

Organizace

- Student absolvuje 3 dvouhodinová cvičení za celý semestr.
- Pro posílení vzájemných vazeb mezi studenty se na cvičení přihlašujete v pětičlenných týmech, což vám umožní chodit na cvičení se spolužáky, se kterými chodíte na cvičení v dalších předmětech (IDM, IEL, ILG a IZP).
- Tvorba týmů i přihlašování na termíny cvičení v IS FIT končí v **neděli 29. září 2019**.

Organizace předmětu – Cvičení (2/2)

Hodnocení

- Za aktivní účast lze na každém cvičení získat **5 bodů**.
- Chyby ani neznalosti ke ztrátám bodů za cvičení nevedou.
- Nečinnost a nezapojení se do cvičení však ano.
- **Nemoc** či jinou překážku lze řešit se cvičícím
  - účastí na jiném cvičení nebo
  - při ohlášení na Studijní oddělení náhradním úkolem.

Asistenti pro cvičení i pro projekt

- doc. Ing. Vladimír Janoušek, Ph.D.
- Ing. Viktor Malík
- Ing. Štefan Martiček
- doc. Mgr. Adam Rogalewicz, Ph.D. – vedoucí
- Ing. Ondřej Valeš
- Ing. Pavol Vargovčík

Organizace předmětu – Hodnocení

- Cvičení: 15 bodů
- Projekt: 25 bodů
- Zápočet: *40 bodů* (min. 18 bodů, tj. 45 %)
- Zkouška: 60 bodů (min. 30 bodů, tj. 50 %)
- Celkem: *100 bodů* (min. 50 bodů, tj. 50 %)
- Pro přistoupení ke zkoušce je nutný zápočet.
- Zřejmě opět vypíšeme 5 termínů zkoušky.
- Zkoušku lze 2× opakovat (jen při neúspěchu u předchozího termínu).

Bodů		Klasifikace	Číselně	Slovně
90 - 100	⇒	A	1	výborně
80 - 89	⇒	B	1,5	velmi dobře
70 - 79	⇒	C	2	dobře
60 - 69	⇒	D	2,5	uspokojivě
50 - 59	⇒	E	3	dostatečně
0 - 49	⇒	F	4	nevyhovující

Organizace předmětu – Projekt

Téma: Model informačního systému (25 bodů)

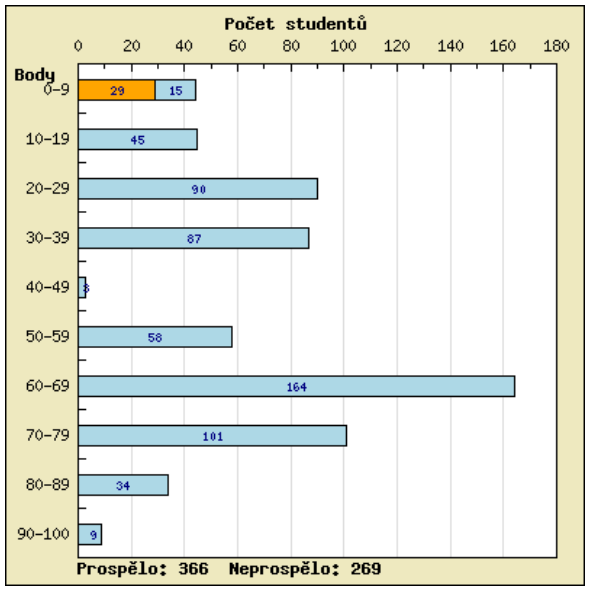
- Diagram případů užití (Use Case Diagram) – požadavky
  - Probírá se na 2. přednášce a 1. cvičení.
- ER diagram (Entity Relationship Diagram) – uchovávaná data
  - Probírá se na 3. přednášce a 2. cvičení.

Termíny

- *Pro uznání bodů se do 6. 10. 2019 přihlaste v IS FIT (jen opakující).*
- Přihlášení na jednu z cca 40 variant zadání v IS FIT
  od pondělí 7. 10. 2019, 20:09 do neděle 3. 11. 2019
- Konzultace s asistentem, který danou variantu zadal.
  Řešení ani konzultace nenechávejte na poslední chvíli.
- Odevzdání do IS FIT ... do neděle 1. 12. 2019
- Obhajoba projektu ... 11. a 12. týden výuky

Projekty vypracovávejte samostatně!

Histogram hodnocení 2016/2017



úspěšnost 60,4 %
(bez neaktivních)

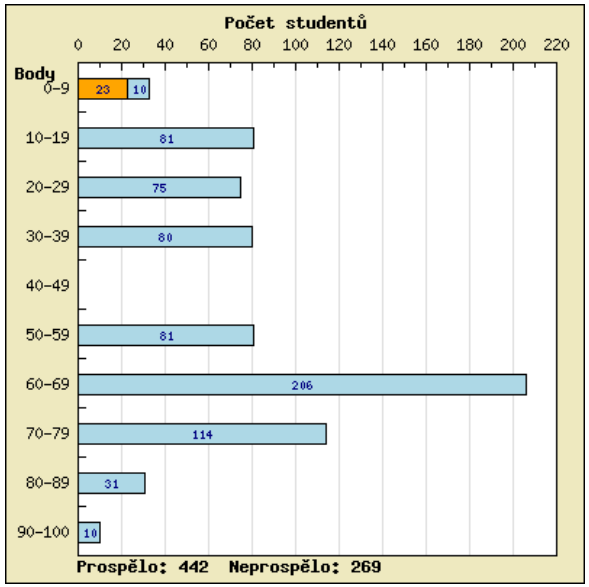
Organizace předmětu – Zkouška

- Studium je investice do vzdělání, která má vysokou návratnost.
  lepší pozice s vyšším platem
- Je především na Vás, jak bude investice 3-5 let života úspěšná. FIT nabízí v praxi vysoce ceněné vzdělání, ale nemůže ho studentům vnutit proti jejich vůli. Snaží se ale chránit své dobré jméno a atraktivitu svých absolventů na trhu práce.
- *Zkouškou se zjišťuje komplexní zvládnutí látky vymezené v dokumentaci předmětu prezentované ve výuce na úrovni odpovídající absolvované části studia a schopnosti získané poznatky samostatně a tvůrčím způsobem aplikovat.* (SZŘ VUT čl. 13 odst. 2)
- **Pro získání bodů ze zkoušky je nutné zkoušku vypracovat tak, aby byla hodnocena nejméně 30 body (ze 60). V opačném případě bude zkouška hodnocena 0 body.**
- Nezkoumejte, jak projít studiem s co nejmenším úsilím, ale sami se snažte naučit co nejvíce. Ovlivní to celou Vaši profesní kariéru!

Organizace předmětu – Komunikace

- **Informační systém fakulty (IS FIT)**
  - termíny a jejich hodnocení
  - diskuzní fóra
  - slidy k přednáškám
  - `https://wis.fit.vutbr.cz/FIT/`
- **Webová stránka předmětu**
  - plán přednášek
  - zadání projektu
  - odkazy na literaturu a další studijní materiály
  - `https://www.fit.vutbr.cz/study/courses/IUS/private/`

Histogram hodnocení 2017/2018

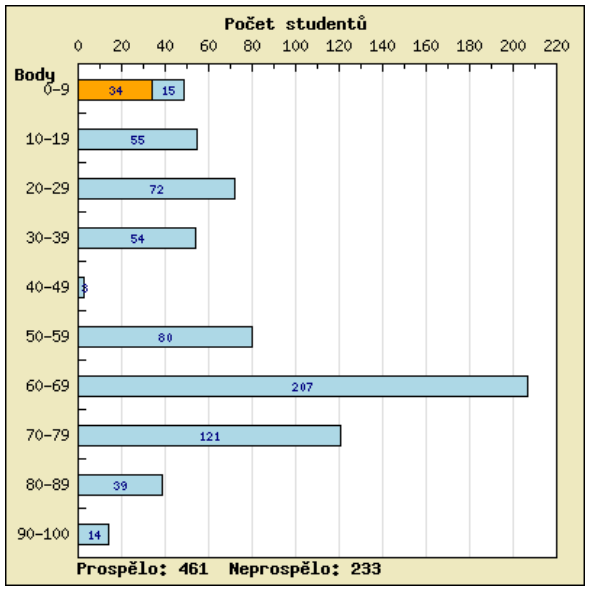


úspěšnost 64,2 %
(bez neaktivních)

Organizace předmětu – Návaznosti

- **Předmět Databázové systémy (IDS)**
  - ER diagramy pro návrh databáze
  - implementace projektu (IS) podle projektu z IUS
- **Ostatní předměty**
  - diagramy UML pro návrh
  - řízení týmových projektů
  - problémy s časem při dokončování projektů
- **Státní závěrečná zkouška – tematické okruhy**
  - 30. Životní cyklus softwaru (charakteristika etap a základních modelů)
  - 31. Jazyk UML
  - 32. Konceptuální modelování a návrh relační databáze
  - 36. Objektová orientace (základní koncepty, třídně a prototypově orientované jazyky, OO přístup k tvorbě SW)
  - `https://www.fit.vut.cz/fit/info/rd/2018/rd36-181123.pdf`
- **Praxe**

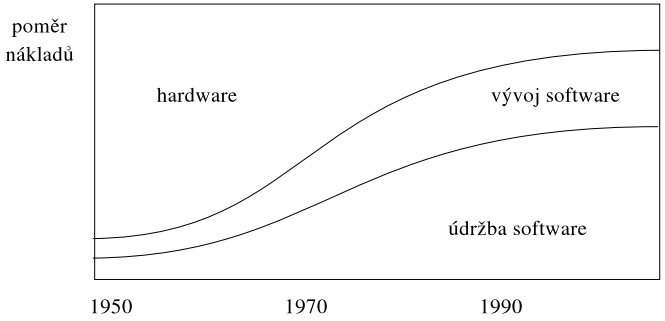
Histogram hodnocení 2018/2019



úspěšnost 69,8 %
(bez neaktivních)

Proč softwarové inženýrství?

- Proč vytváříme software?
  - zlepšení služeb – informační systémy, ...
  - snížení nákladů – řízení výroby, ...
  - nemožnost řešení bez použití počítačů – předpověď počasí, ...
- Je nutné zlepšovat vlastnosti SW, hlavně jeho spolehlivost, bezpečnost a použitelnost.
- Je potřeba zvyšovat produktivitu vývoje SW.



Cíle předmětu

- získat základní přehled v oblasti tvorby rozsáhlých softwarových systémů,
- seznámit se s procesem tvorby softwaru a s etapami jeho životního cyklu,
- naučit se používat základní modely UML.

Proč softwarové inženýrství?

Katastrofy (málem) způsobené SW chybou

- 1996: Přetečení při konverzi 64 b. čísla v plovoucí řádové čárce na 16 b. celé číslo se znaménkem reprezentující vertikální rychlost vedlo 40 s po startu k autodestrukci rakety Ariane 5.
- 1985-1987: V důsledku odstranění hardwarové zábrany proti nadměrnému ozáření při vývoji lékařského přístroje Therac-25 a SW chyb bylo nadměrně ozářeno 6 pacientů (3 na následky zemřeli). Varující je zejména přístup výrobce, který při prvních případech nadměrného ozáření místo nápravy tvrdil, že k němu **nemůže** dojít.
- 1983: Sovětský systém pro včasné varování před nukleárním útokem nahlásil obsluze pět balistických střel mířících z USA na Moskvu. Operátor naštěstí použil hlavu (pokud by USA zaútočily na Sovětský svaz, použily by více než pět raket) a vyhodnotil to jako falešný poplach (odrazy slunce od mraků) místo odvety.
- 2018/2019: Boeing 737 MAX 8
- Další informace jsou např. na URL:
  `http://www5.in.tum.de/~huckle/bugse.html`

Co je to softwarové inženýrství?

- systematický přístup k vývoji, nasazení a údržbě softwaru
  The application of a systematic, disciplined, quantifiable approach to development, operation, and maintenance of software; that is, the application of engineering to software. (IEEE Standard Computer Dictionary, 1990)
- inženýrská disciplína zabývající se praktickými problémy vývoje rozsáhlých softwarových systémů (Vondrák, 2002)

! softwarové inženýrství $\neq$ programování

(Ne)úspěšnost SW projektů (Standish Group Report, USA, 1995)

Výsledná funkčnost	Projektů
méně než 25 %	4,6 %
25 - 49 %	27,2 %
50 - 74 %	21,8 %
75 - 99 %	39,1 %
100 %	**7,3 %**

Průměrný SW projekt tedy v porovnání s původním plánem:

- stál o **89 %** více,
- trval **2,22 krát** déle a
- poskytuje pouze **61 %** funkčnosti.

Průměrný projekt byl tedy téměř 7 krát horší, než se původně plánovalo!

Počátek SW inženýrství

Počátek – šedesátá léta 20. století

- problémy při vývoji větších programů
- zavedení pojmů *softwarové inženýrství* a *softwarová krize* na konferencích v letech 1968-1969
- SW krize se projevovala (a stále projevuje)
  - neúnosným prodlužováním a prodražováním projektů
  - nízkou kvalitou výsledných produktů
  - problematickou údržbou a inovacemi
  - špatnou produktivitou práce programátorů
  - řada projektů končila neúspěchem
- první kroky k metodickému přístupu k programování – strukturované programování

Problémy při vývoji softwaru

Podstatné, vnitřní, nevyhnutelné problémy:

- **Složitost** – žádné dvě části nejsou stejné; složitost je zdrojem dalších problémů jako např. komunikace v týmech; je náročné pochopit všechny možné stavy systému; problémy s úpravami a rozšířeními, ...
- **Přizpůsobivost** – když se něco změní, měl by se přizpůsobit software a ne naopak.
- **Nestálost** – mění se okolí a mění se i software (nejde o nahrazení novým); přibývají požadavky na úspěšně používaný software; software přežívá hardwarové prostředky.
- **Neviditelnost** – neexistuje přijatelný způsob reprezentace softwarového výrobku, který by pokryl všechny aspekty; dokonce ani nejsme schopni určit, co v dané reprezentaci chybí.

  Syndrom *90% hotovo*: Při posuzování hotové části se nevychází z hotového, ale z odpracovaného (např. podle plánu).

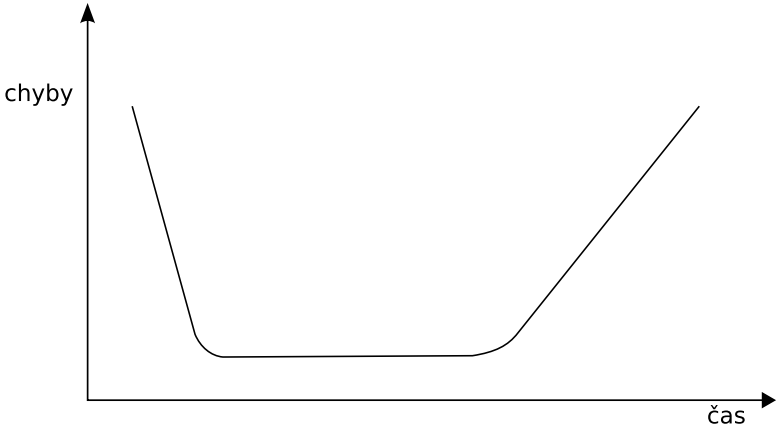
(Ne)úspěšnost SW projektů (Standish Group Report, USA, 1995)

Překročení nákadů o	Projektů
méně než 20 %	**15,5 %**
21 - 50 %	31,5 %
51 - 100 %	29,6 %
101 - 200 %	10,2 %
201 - 400 %	8,8 %
více než 400 %	4,4 %

Překročení času o	Projektů
méně než 20 %	**13,9 %**
21 - 50 %	18,3 %
51 - 100 %	20,0 %
101 - 200 %	35,5 %
201 - 400 %	11,2 %
více než 400 %	1,1 %

Stárnutí hardwaru

Typická chybová křivka hardwaru



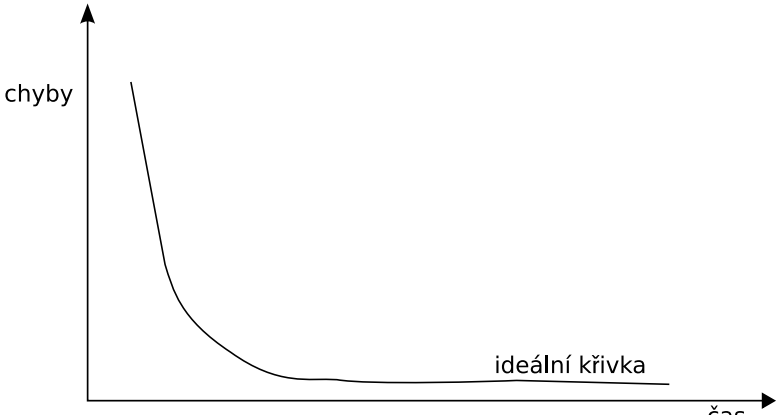
Problémy při vývoji softwaru

Problémy, které se nemusí projevit vždy:

- **práce v týmu**
  - problémy s organizací práce na velkých softwarových projektech
  - problémy s plánováním procesu tvorby softwaru
  - Komunikační problémy jsou jedním z hlavních zdrojů chyb v programech.
  - extrémní odchylky v produktivitě mezi jednotlivými programátory, až 1:20
- **nízká znovupoužitelnost při tvorbě softwaru**
  - V procesu tvorby softwaru je málo standardů a většinou se software tvoří od začátku. S každým programem se vymýšlí už vymyšlené.
  - Málo produktů se sestavuje z už existujících součástí.
- **problém míry**
  - Metody použitelné na řešení malých problémů se nedají přizpůsobit na řešení velkých (složitých) problémů.

Stárnutí softwaru

Typická chybová křivka softwaru



Problémy při vývoji softwaru

Problémy, které se nemusí projevit vždy:

- **tvorba dokumentace**
  - Tvorba dokumentace je podobná tvorbě vlastního programu.
  - enormní rozsah dokumentace co do kvantity i rozmanitosti
    Např. ve velkých vojenských softwarových projektech připadalo 400 anglických slov na každý příkaz v programovacím jazyce Ada.
  - problémy s udržováním aktuálnosti dokumentace vzhledem ke změnám softwaru
  - problémy s konzistencí a úplností dokumentace
- **náchylnost softwaru k chybám**
  - Hodně chyb se projeví až při provozu (a ne při vývoji).
  - Odstraňování chyb vede k návratu v etapách vývoje softwaru.
- **způsob stárnutí softwaru**
  - Software se fyzicky neopotřebuje. ALE: Přidávání nových funkcí ve spojení s častými opravami chyb vede k postupné degradaci struktury a k snižování spolehlivosti softwarových systémů.

Příčiny zastavení softwarových projektů

. . . podle analýzy víc jak 350 firem a 8000 aplikací:

- neúplnost nebo nejasnost požadavků (13,1 %)
- nedostatek zájmu a podpory ze strany uživatele (12,4 %)
- nedostatek zdrojů, tj. podhodnocený rozpočet a krátké termíny (10,6 %)
- nerealistické očekávání (9,9 %)
- malá podpora od vedení dodavatele nebo odběratele (9,3 %)
- změna požadavků a specifikace (8,7 %)
- nedostatečné plánování (8,1 %)
- vyvíjený systém už není potřeba (7,5 %)
- . . .

Problémy při vývoji softwaru

Problémy, které se nemusí projevit vždy:

- **specifikace požadavků**
  - problematická komunikace s uživatelem
  - nejasná a neúplná formulace požadavků spojená s neucelenou představou uživatele o výsledném softwarovém systému
  - nejednoznačnost spojená s častou specifikací požadavků v přirozeném jazyce
  - . . .

Tvorba softwaru je tvůrčí proces, software nelze vyrábět.

Pár postřehů Freda Brookse

- Přidáním dalších pracovníků do zpožděného projektu se tento projekt ještě více zpozdí.
- Napsání překladače Algolu zabere 6 měsíců nezávisle na tom, kolik ho vytváří programátorů.
- Efekt (syndrom) druhého systému – při návrhu druhé verze systému hrozí rizika:
  - příliš složitý a neefektivní systém

    Systém není dokonalý, když k němu nelze nic přidat, ale tehdy, když z něho nelze nic odstranit.
  - nepoužití nových technologií

Problémy při vývoji softwaru

Příklad důsledku nepřesnosti či nepochopení specifikace.

Personání oddělení (PO): "Máme problém se systémem. Zaměstnankyně změnila jméno a systém změnu neakceptuje."

IT oddělení (IT): "Provdala se?"

PO: "Ne, pouze změnila jméno. Systém zřejmě vyžaduje změnu stavu osoby."

IT: "Ano, nikdy jsme neuvažovali, že by si někdo změnil jméno jen tak."

PO: "Předpokládali jsme, že víte, že lidé mohou kdykoliv legálně změnit jméno. Potřebujeme změnu jména zavést do systému, abychom mohli zadat výplatu. Kdy odstraníte chybu?"

IT: "To není chyba! Nevěděli jsme, že potřebujete tuto vlastnost. Můžeme tuto novou vlastnost zavést do konce měsíce. Příště nám své požadavky řekněte dříve."

K. Wiegers, J. Beatty: Software Requirements. Microsoft Press, 2013.

Softwarový produkt

Program

- funkční část produktu

Softwarový produkt

- sbírka počítačových programů, procedur, pravidel a s nimi spojená dokumentace
- zahrnuje např.: požadavky, specifikace, popisy návrhu, zdrojové texty, testovací data, příručky, ...

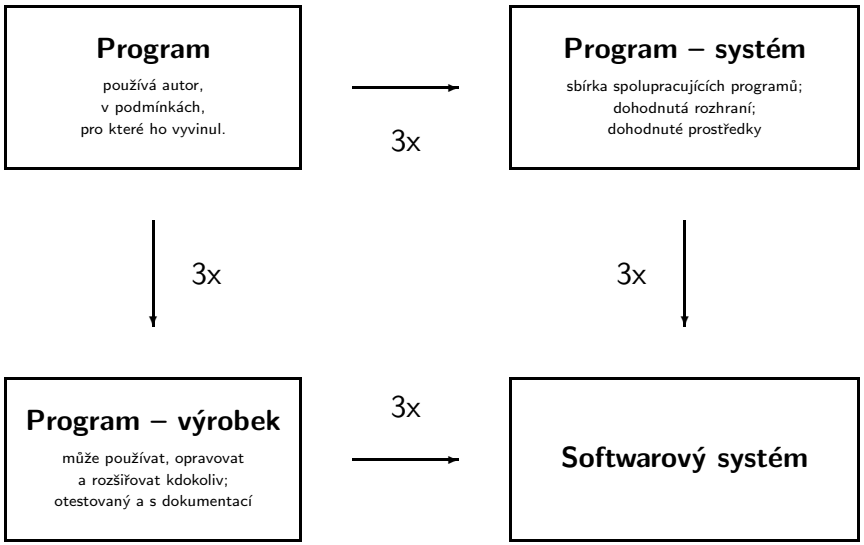
Aktéři ve vývoji softwarového produktu (softwaru)

- **Zákazník** – sponzoruje vývoj SW, specifikuje požadavky na SW
- **Dodavatel** – vyvíjí systém, má závazky vůči zákazníkovi, komunikuje s uživatelem (testování, ...)
- **Uživatel** – testuje a používá systém, upřesňuje požadavky na SW

Rozvoj SW inženýrství

- Výzkum programovacích praktit
  - uvědomění si lidského faktoru, práce v týmu
  - podpora řízení tvorby SW
  - modulární programování
  - návrhové vzory
- Výzkum metodik
  - vnímání životního cyklu vývoje SW
  - strukturované metody, datově a procesně orientované metody, objektově orientované metody, agilní metodiky, ...
  - výzkum modelovacích jazyků (dnes UML)
- Zabezpečení kvality
  - systematické testování, formální ověřování
- Metody návrhu založené na modelech
  - transformace modelů do programu

Vztah mezi programem a softwarem



Metodiky vývoje softwaru

Metodiky

- disciplinovaný proces nad vývojem softwaru s cílem zajistit tento vývoj více predikovatelný a efektivnější
- věnují se různým aspektům, které ovlivňují vývoj softwarového produktu, včetně samotného procesu tvorby softwaru
- zahrnují proces vývoje, nástroje, způsoby využití, plánování, ...

Pozor na terminologii!

- *Metoda* – postup pro dosažení určitého cíle
- *Metodika* – souhrn doporučených praktik a postupů
- *Metodologie* – nauka o metodách, jejich tvorbě a použití

Ale!

Metodika vývoje softwaru = Software Development Methodology

Proces vývoje softwaru

Proces, ve kterém

- se potřeby uživatele transformují na požadavky na SW,
- požadavky na SW se transformují na návrh,
- návrh se implementuje,
- implementace se testuje
- a nakonec předá uživateli.

SW proces definuje

- kdo
- dělá co
- a kdy
- $\Rightarrow$ jak dosáhnout požadovaného cíle

Typy softwarových produktů

Generické

- Software se prodává libovolnému zájemci (krabicový software).
- Musí být velice důkladně otestován, protože opravy chyb jsou vzhledem k velkému rozšíření drahé.

Zákaznické (na objednávku)

- Software se vytváří na základě požadavků pro konkrétního zákazníka.
- Většinou pro specializované aplikace, pro které vhodný generický software neexistuje.
- Cena zákaznického softwaru je výrazně vyšší.
- Dvě možnosti jeho tvorby:
  - zadáním zakázky SW firmě
  - v rámci vlastní firmy

Životní cyklus softwaru

Životní cyklus

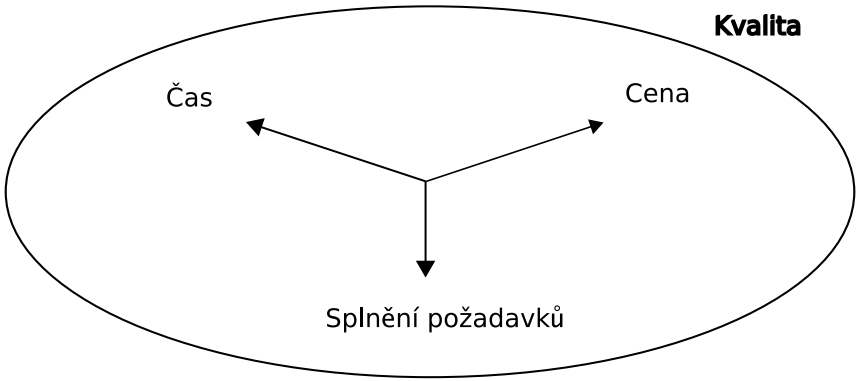
- rozděluje proces vývoje softwaru na za sebou jdoucí období
- pro každé období stanovuje cíl
- období = **etapa životního cyklu softwaru**

Činnosti spojené s vývojem softwaru

- analýza a specifikace požadavků (8 %),
- architektonický a podrobný návrh (7 %),
- implementace (12 %),
- integrace a testování (6 %),
- provoz a údržba (67 %).

Úsilí věnované pečlivé analýze a návrhu se vrátí úsporou nákladů později.

Kvalita SW produktů



Dekompozice složitých problémů

Přináší

- lépe zvládnutelné podsystémy
- soustředění pozornosti na jeden podsystém
- prezentovatelnost dílčího problému bez rušivých vlivů
- podsystémy se mohou vyvíjet nezávisle
- **skutečně velké systémy se bez dekompozice nedají zvládnout**

Zvýšená pozornost

- koordinace tvorby rozhraní
- integrace a testování podsystémů

Etapy životního cyklu softwaru

Analýza a specifikace požadavků

- získávání, analýza, definování a specifikace požadavků ⇒ transformace neformálních požadavků uživatele do strukturovaného popisu požadavků,
- cílem je identifikovat požadavky uživatele, ne návrh, jak je realizovat,
- provedení studie vhodnosti, identifikace a analýza rizik,
- plánování akceptačního testování.

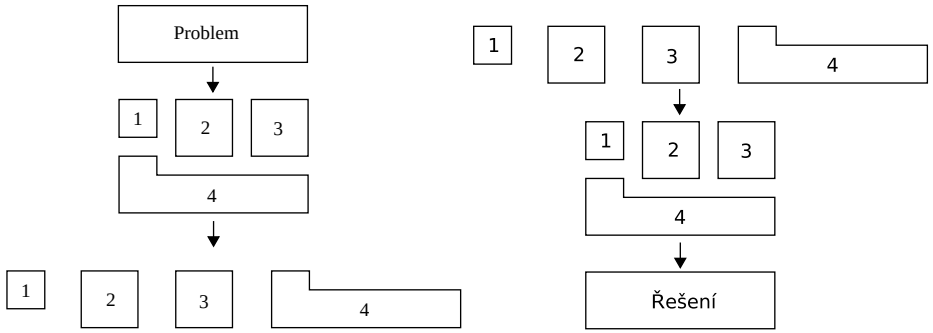
Etapy životního cyklu softwaru

Architektonický návrh

- ujasnění koncepce systému,
- dekompozice systému,
- definování vztahů mezi částmi systému,
- specifikace funkcionality a ohraničení podsystémů,
- plánování testování systému,
- plánování nasazení systému do provozu, dohoda o postupu nasazování podsystémů, dohoda o plánu zaškolování uživatelů.

Dekompozice složitých problémů

- rozdělení (dekompozice) složitějšího problému na jednodušší (lehčí zvládnutí problému)
- rozhraní podsystémů



Etapy životního cyklu softwaru

Akceptační testování a instalace

- testování systému uživatelem,
- operace přebírání SW produktu,
- školení používání systému, nasazení systému.

Provoz a údržba

- zabezpečení provozu softwaru,
- řešení problémů s nasazením softwaru,
- řešení problémů s používaním softwaru,
- opravy, rozšiřování, přizpůsobování softwaru podle požadavků okolí.

Etapy životního cyklu softwaru

Podrobný návrh

- podrobná specifikace softwarových součástí,
- specifikace algoritmů realizujících požadované funkce,
- specifikace rozhraní pro jednotlivé součásti,
- specifikace logické a fyzické struktury údajů,
  které zpracovává příslušná součást,
- specifikace způsobu ošetřování chybových a neočekávaných stavů,
- plán prací při implementaci součásti,
- plán testování součásti, návrh testovacích dat,
- specifikace požadavků na lidské zdroje (odhad trvání a nákladů projektu).

Model životního cyklu softwaru

Model životního cyklu

- definuje etapy vývoje softwaru a jejich časovou následnost,
- pro každou etapu definuje nutné činnosti,
- pro každou etapu definuje její vstupy a výstupy.

Další vlastnosti

- nedefinuje délku trvání kroků a jejich rozsah,
- každá etapa vytváří reálné výstupy,
- správnost každé etapy lze vyhodnotit.

Rozdíly v modelech jsou zejména v definování etap a jejich posloupnosti.

Etapy životního cyklu softwaru

Implementace a testování součástí

- programová realizace softwarových součástí,
- vypracování dokumentace k součástem,
- testování implementovaných součástí,
- začátek školení budoucích uživatelů.

Integrace a testování systému

- spojení součástí do podsystémů,
- testování podsystémů,
- integrace podsystémů do celého systému,
- testování podsystémů a celého systému
  oprava nalezených chyb, návraty k etapě implementace.

Hlavní cíle SW inženýrství

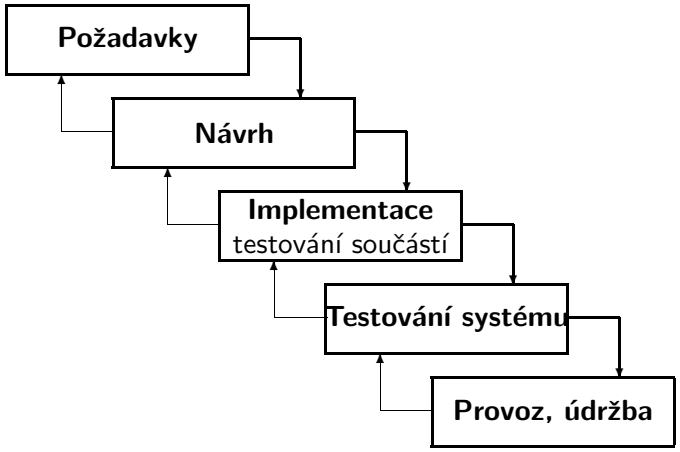
- **Management projektu**
  - řízení životního cyklu projektu
  - dosažení požadovaného výsledku v požadovaném čase
  - $\Rightarrow$ efektivní práce s časem a tedy i s náklady
- **Techniky**
  - analýzy
  - návrhu
  - programování
  - testování
  - ...
- **Vlastnosti SW inženýra**
  - základní báze znalostí
  - schopnost aplikovat znalosti
  - schopnost vyhledávat nové informace a osvojit si nové znalosti
  - ...

Studijní koutek

- Měl by vám pomoci s orientací při studiu.
- Je pro něj vyhrazeno posledních 10 minut přednášky.
- Zde uvedené informace se nezkoušejí.
- Můžete posílat náměty na to, co vás zajímá.

Vodopádový model životního cyklu softwaru

- životní cyklus jde postupně od první etapy až do poslední
- následující etapa začne až po ukončení předcházející
- možnost návratu k předchozí etapě



Vodopádový model

Vlastnosti

- lineární (sekvenční) model, intenzivně používán v 70. letech
- cílem bylo zavést do vývoje řád umožňující řešit náročnější problémy
- dekompozice, kontrola výstupů etap $\Rightarrow$ snížení počtu chyb
- uživatel se účastní pouze při definování požadavků a zavádění

Výhody

- jednoduché na řízení
- při stálých požadavcích: nejlepší struktura výsledného produktu

Nevýhody

- zákazník není schopen předem stanovit (přesně!) všechny požadavky
- při změnách požadavků dlouhá doba realizace
- zákazník vidí spustitelnou verzi až v závěrečných fázích projektu $\Rightarrow$ odhalení nedostatků ve specifikaci požadavků příliš pozdě (validace)

Vysoké učení technické v Brně

- **Historie**
  - 1849 – německo-české technické učiliště
  - **1899** – Česká vysoká škola technická v Brně
  - 1956 – Vysoké učení technické v Brně
- **Vedení**
  - Nejvyšším představitelem vysoké školy je rektor.
  - 52. rektorem je **prof. RNDr. Ing. Petr Štěpánek, CSc., dr. h. c.**
- **Fakulty**
  - Fakulta architektury – FA
  - Fakulta elektrotechniky a komunikačních technologií – FEKT
  - Fakulta chemická – FCH
  - **Fakulta informačních technologií – FIT**
  - Fakulta podnikatelská – FP
  - Fakulta stavební – FAST
  - Fakulta strojního inženýrství – FSI
  - Fakulta výtvarných umění – FAVU

Fakulta informačních technologií

- **Historie**
  - 1964 – Katedra samočinných počítačů na FE
  - 1990 – Katedra informatiky a výpočetní techniky na FE
  - 1992 – Ústav informatiky a výpočetní techniky na FE
  - 1993 – reorganizace FE $\Rightarrow$ FEI
  - 2002 – Fakulta informačních technologií (FIT)
- **Ústavy**
  - Ústav informačních systémů
  - Ústav inteligentních systémů
  - Ústav počítačové grafiky a multimédií
  - Ústav počítačových systémů

Fakulta informačních technologií

- **Vedení**
  - Nejvyšším představitelem fakulty je děkan.
    1. děkanem FIT byl (2002-2008) **prof. Ing. Tomáš Hruška, CSc.**
    2. děkanem FIT byl (2008-2016) **doc. Ing. Jaroslav Zendulka, CSc.**
    3. děkanem FIT je (od 2016) **prof. Dr. Ing. Pavel Zemčík**
  - Proděkanem pro vzdělávací činnost
    v bakalářském studiu je **Ing. Jaroslav Dytrych, Ph.D.**
    v magisterském studiu je **doc. Ing. Richard Růžička, Ph.D., MBA**
  - Studijní poradci jsou
    Ing. Miloš Eysselt, CSc. a
    Ing. Petr Veigend