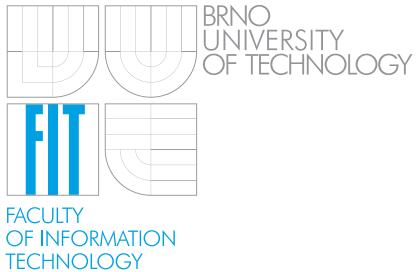
Motivace, obsah přednášek, projekty



Motivace (1)

- V rámci předmětů bakalářského programu budete zpracovávat řadu samostatných i týmových projektů
- V praxi se můžet stát členy velmi malého týmu, případně pracovat zcela samostatně (tvorba webů a další malé projekty)

- Lze vystačit bez znalostí IVS (v roce 2011 první běh kursu)

- Ale nástroje a postupy, s nimiž se seznámíte v IVS, vám mohou pomoci vytvářet systémy udržovatelné po delší době, nikoliv jen v měsíci, kdy vznikly

Motivace (2)

- Jinou možností je, že se v rámci nového zaměstnání setkáte s vágně pojatým procesem vývoje software; volnými či neexistujícími normami a změtí nekompatibilních nástrojů
- IVS by vás mělo naučit, které nástroje a postupy se vyplatí zavést

- Je také možné, že se v praxi setkáte s dobře definovaným procesem vývoje, normami a sdílenou sadou nástrojů
- Musíte být schopni pracovat v takovém prostředí a rychle si osvojit pojetí dané specifickou koncepcí vývoje
- V IVS byste měli získat zkušenost s typickými postupy, což by mělo pomoci ke snazšímu pochopení specifik nových nástrojů

Krize? vývoje software

Panuje obecná shoda, že většina velkých softwarových projektů končí neúspěšně (není dokončena včas, náklady vývoje výrazně překračují rozpočet a výsledek neodpovídá rozsahem nebo kvalitou potřebám uživatelů)

Standish project benchmarks over the years

Year	Successful (%)	Challenged (%)	Failed (%)
1994	16	53	31
1996	27	33	40
1998	26	46	28
2000	28	49	23
2004	29	53	18
2006	35	46	19
2009	32	44	24

Co se v předmětu nedozvíte

- Jak se vyhnout výše uvedeným problémům – minimálně od 60. let 20. století se o to snaží oblast softwarového inženýrství (viz předmět IRP)
- Jak se naučit programovat v konkrétním programovacím jazyce / používat vývojové nástroje specifické pro konkrétní jazyk (viz např. předmět ISJ)
- Teoretické základy, na nichž stojí různé nástroje (např. pro odhalování chyb) – takových předmětů je na FIT hodně
- Vyčerpávající přehled dostupných nástrojů, s nimiž se lze v praxi setkat

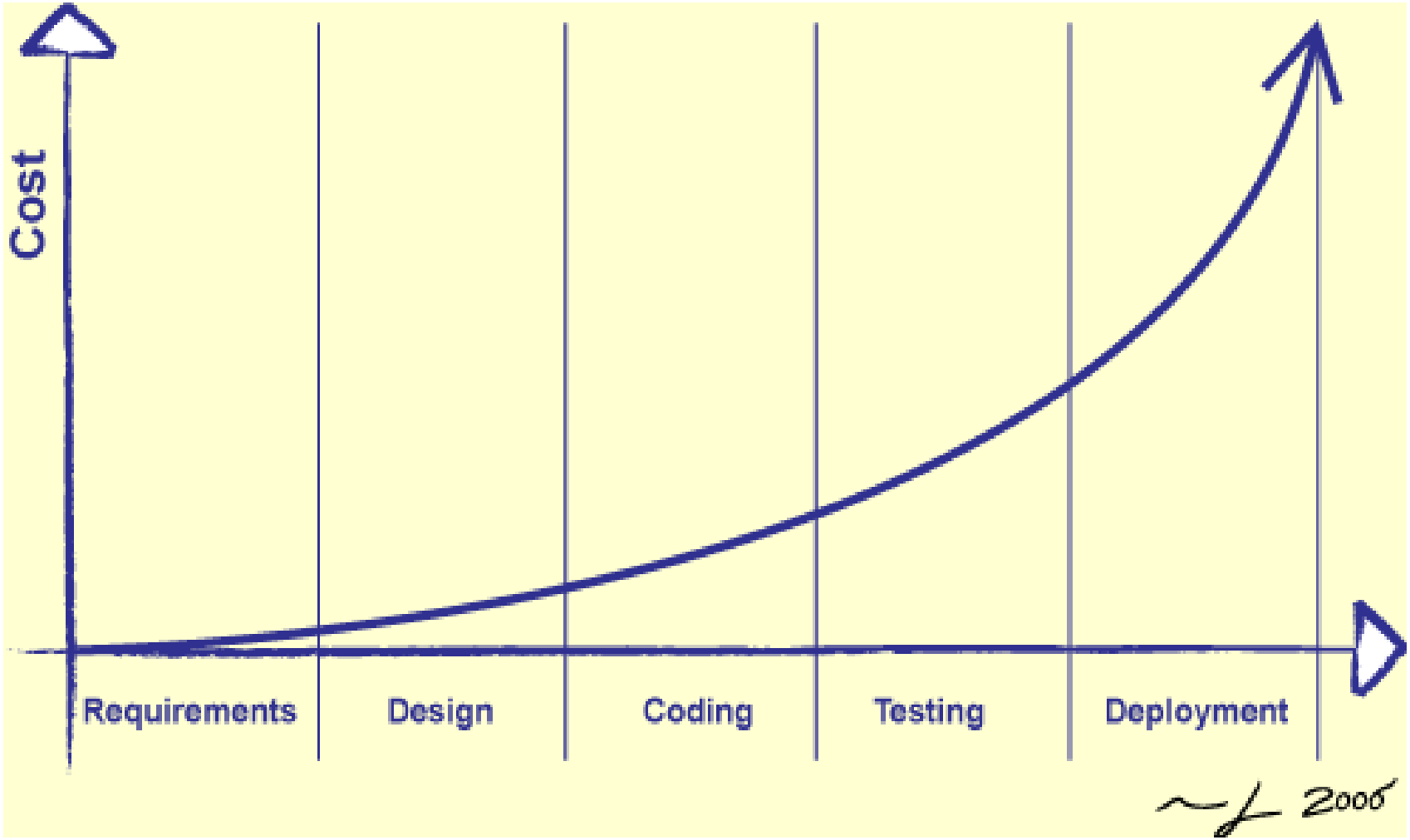
Přehled přednášek

- Praktická aplikace unixové filosofie při programování
- Role testů při vývoji software
- Odhalování chyb, profiling, debugging
- Identifikace existujících komponent, využívání knihoven dostupných na různých platformách, programy pro sestavování jako samodokumentující prostředí
- Generování programové dokumentace z kódu
- Systémy pro distribuovanou správu verzí, GIT
- Návrh uživatelských rozhraní, zážitky uživatelů
- Měření upotřebitelnosti
- Urychlování výpočtů, možnosti paralelizace
- Využívání alternativních paradigmat programování
- Přednáška pozvaných expertů z firemní praxe

- Většina moderních nástrojů má grafické uživatelské rozhraní (GUI)
  - Použití je jednodušší
  - Obrázek lepší než tisíc slov
- Ale rozhraní příkazové řádky mají stále svoje místo
  - Jednodušší implementace
  - Vyšší poměr akce na jedno stisknutí klávesy
  - Snadněji pochopitelné, co program dělá za člověka
  - Jednodušší kombinování nástrojů mnoha způsoby

Návrh versus pohotovost/hbitost

Barry W. Boehm – Software Engineering Economics (1981)



Návrh versus pohotovost/hbitost

Současné debatě o vývoji software dominují dvě školy:

- BDFU (Big Design Up Front) - dvakrát měř, jednou programuj
  - dobře promysli potřeby uživatelů, návrh a možné problémy před tím, než začneš programovat
- Agilní přístup – mnoho malých krůčků, nepřetržité testování a restrukturalizace kódu
- Oba přístupy se snaží vypořádat s Boehmovou křivkou:
  - BDFU – snaha vyhnout se zcela problémům – The cheapest bug to fix is one that doesn't exist
  - Agilní – snaha zachytit problémy ještě ve fázi, kdy je cena nízká
- V praxi jsou rozdíly mnohem menší než je obvykle deklarováno

Test-first

- Testy jsou psány před psaním vlastního kódu
- Pokud existuje test, je jednodušší a rychlejší napsat kód
- Napsání testu pomáhá k ujasnění, co má být skutečně uděláno – jsou vyjasněny požadavky, omezí se pozdější nedorozumění
- Okamžitá zpětná vazba – bez testů často není jasné, kdy je dokončena veškerá požadovaná funkcionality (rozšiřování, ošetření dalších chybových stavů atd.)
- Lepší návrh systému – pokud nejprve vznikl složitý kód, často je obtížné vytvořit testy programových jednotek (navíc může jít o zcela jiný tým). Pokud vytváříte nejprve testy, návrh systému bude ovlivněn snahou otestovat vše významné
- Ostatní mohou snadno pochopit fungování kódu procházením testů
- Správné testy programových jednotek běží rychle (krátký čas pro nastavování, čas vlastního běhu a vyhodnocení chyb), nezávisle na ostatních (možnost přeuspořádání), data přispívají k jednoduchosti a pochopení, pokud možno reálná

TDD = Test-first + Refactoring

Refactoring – ukázněná změna struktury kódu – malé změny vedoucí k lepšímu kódu (jednoduššímu k pochopení a k pozdějším úpravám), umožňuje postupné změny (iterative and incremental) – např. prosté přejmenování metody, nikoliv přidání funkcionality

1. Piš kód, pouze pokud neuspěl některý test
2. Odstraň duplicity, které jsi našel

Kent Beck vysvětlil, jak tato 2 jednoduchá pravidla ovlivní chování jednotlivců a týmů

- Vývojáři píší vlastní testy, protože nemají čas čekat 20 minut denně na někoho, kdo je napíše za ně
- Vývojové prostředí musí mít rychlou odezvu na malé změny (rychlá kompilace a systém pro vyhodnocení všech testů)
- Návrh musí sestávat z kompaktních, volně provázaných částí (většinou je normalizován), aby se testy zjednodušily (důsledkem je také snazší rozšiřování a údržba)

Typy nástrojů využívaných při vývoji software

- Sestavování částí
- Správa verzí
- Odhalování chyb
- Testování
- Evidování chyb a námětů
- Generování dokumentace
- Řízení projektů
- Integrace

Přehled přednášek

- Praktická aplikace unixové filosofie při programování
- Role testů při vývoji software
- Odhalování chyb, profiling, debugging
- Identifikace existujících komponent, využívání knihoven dostupných na různých platformách, programy pro sestavování jako samodokumentující prostředí
- Generování programové dokumentace z kódu
- Systémy pro distribuovanou správu verzí, GIT
- Návrh uživatelských rozhraní, zážitky uživatelů
- Měření upotřebitelnosti
- Urychlování výpočtů, možnosti paralelizace
- Využívání alternativních paradigmat programování
- Přednáška pozvaných expertů z firemní praxe