Řízení reálných projektů, agilní metodiky

Antonín Komenda

Agent Technology Group
Katedra kybernetiky
Fakulta elektrotechnická - České vysoké učení technické

Praha, 2009

Osnova

- menší komerční firmy (zejména vývoj web aplikací)
- akademické prostředí (vývoj Java aplikací)

1. Lze vyvíjet software bez metodiky? - bohužel ano
2. Klasické metodiky jsou příliš kostnaté? - použijme agilní
3. Vývoj frameworků / knihoven? - zcela jiný příběh

Lze vyvíjet software bez metodiky? - bohužel ano

Popis reálné zkušenosti s vývojem CMS systému

Joelův test:

- **Používáte systém pro správu verzí?**
- Dokážete udělat build v jednom kroku?
- Děláte každodenní buildy?
- **Máte databázi chyb?**
- Opravujete chyby předtím, než začnete psát nový kód?
- Máte průběžně aktualizovaný plán prací?
- Píšete specifikace?
- Mají programátoři klidné pracovní prostředí?
- Používáte ty nejlepší nástroje, které jsou k dispozici?
- **Máte testery?**
- Píší uchazeči o práci kód během přijímacího pohovoru?
- Děláte "pouliční" testy uživatelského rozhraní?

Nekaždý vedoucí chápe rozdíl mezi výrobou softwaru a výrobou rohlíků

Odhady časů:

- kvalitní práce potřebuje čas, nekvalitní práce potřebuje ve finále času x-krát tolik
- není absolutně možné odhadovat časy nad kódem, který je vyvíjen bez testingu
- metriky hodnocení programátorů

Kvality kódu:

- práce jednoho člověka vs. práce ve více lidech
- zdrojové kódy jsou pro programátory na prvním místě, potom pro kompilátor
- unit testing, interface testing, module testing
- modularita
  - hierarchie
  - volné vazby (loosely coupling)
  - žádná duplikace kódu (vs. copy&paste programming)
- coding style (dobré mravy zápisu kódu)

"Any fool can write code that a computer can understand.
Good programmers write code that humans can understand."

— Martin Fowler, Refactoring: Improving the Design of Existing Code

Klasické metodiky jsou příliš kostnaté? - použijme agilní

Klasické metodiky vs. agilní metodiky vývoje softwaru

Agilní metodiky:

- důraz na průběžnou komunikaci mezi vývojovým týmem a zákazníkem
- důraz na tvorbu kvalitního kódu a funkcí, které mají přímou obchodní hodnotu pro zákazníka
- týmovou spolupráci a samoorganizaci týmů
- co nejčastější předávání hotové práce
- vítání změn jako příležitosti být lepší
- důraz na výslednou hodnotu pro zákazníka před dokumenty a papíry

Agilní metodiky:

- běžné aplikace
- zkušení vývojáři
- požadavky se velmi často mění
- malý počet vývojářů
- založeno na principech volné spolupráce

Klasické metodiky:

- kritické aplikace
- nezkušení vývojáři
- požadavky se nemění (příliš často)
- velký počty vývojářů
- založeno na principech pevného řádu

Příklady agilních metodik: Extrémní programování, SCRUM, Crystal Clear, Adaptivní vývoj, . . .

Feature Driven Development

- vývoj řízený aktuálními požadavky na funkcionality
- vývoj probíhá v iteracích
  - soupis všech požadavků na funkcionality
  - analýza funkcionalit a odhad času vývoje a ceny
  - přidělení priorit funkcionalitám zákazníkem
  - vlastní implementace vybraných funkcionalit
  - testování
  - nasazení (deployment)
- změny by měl zákazník aktivně sledovat a komentovat
- zákazník může operativně zasahovat do vývoje

Test Driven Development

- napiš (rozšiř) automatický test nově požadované funkcionality
- naimplementuj aby všechny testy prošly
- refaktorizuj (odstraň duplicitu v kódu)
- opakuj proces

Popis reálné zkušenosti s vývojem informačního systému

Vývoj frameworků / knihoven? - zcela jiný příběh

Žádný software se dnes nepíše od nuly:

- knihovna (balík znovupoužitelných kódů)
- framework „rámcový systém” (software zjednodušující vývoj aplikací určitého druhu)
- toolkit (balík nástrojů)

Volba vhodných technologií:

- co znám
- co splňuje moje požadavky
- co je dobře zdokumentované
- co když technologii musí používat více lidí?

Každý vývojář jednou bude používat nějaký framework / knihovny:

- firemní (in-house)
- volba z existujících (open source, proprietární)
- napsat si sám

Dobře si rozmyslete vývoj frameworku:

- vývoj FW je velmi časově / finančně náročný
- návrh / vývoj kvalitního FW potřebuje zkušené vývojáře
- FW se musí udržovat
- FW musí být špičkově dokumentován
- FW musí mít pochopitelné API
- když už je nutné framework vyvinout
  - zveřejněte ho pod nějakou open source licencí
  - komerční FW musí mít precizně provedenou analýzu prodeje

Popis reálné zkušenosti s vývojem vizualizačního frameworku

Závěr

- vyvíjet software může dnes skoro každý, vyvinout ale úspěšně fungující software není vůbec jednoduché

Reference:

- Joel Spolsky, Joelův test, online: http://czech.joelonsoftware.com/Articles/TheJoelTest.html
- Agile Software Development, online: http://en.wikipedia.org/wiki/Agile_software_development
- 97 Things Every Software Architect Should Know - The Book: http://97-things.near-time.net/wiki/97-things-every-software-architect-should-know-the-book

Otevřená diskuse

- s jakými problémy se se při vývoji studenti nejvíce setkávají?
- nebylo by fajn, vyzkoušet některé principy už při při psaní semestrálních prací?
  - verzovací systém
  - správa chyb
  - iterace přes trojici "analýza -> implementace -> testování"
  - čistota kódu
- píšeme kódy, kterým za pár let porozumíme?

Otevřená diskuse

- s jakými problémy se se při vývoji studenti nejvíce setkávají?
- nebylo by fajn, vyzkoušet některé principy už při při psaní semestrálních prací?
  - verzovací systém
  - správa chyb
  - iterace přes trojici "analýza -> implementace -> testování"
  - čistota kódu
- píšeme kódy, kterým za pár let porozumíme?

Otevřená diskuse

- s jakými problémy se se při vývoji studenti nejvíce setkávají?
- nebylo by fajn, vyzkoušet některé principy už při při psaní semestrálních prací?
  - verzovací systém
  - správa chyb
  - iterace přes trojici "analýza -> implementace -> testování"
  - čistota kódu
- píšeme kódy, kterým za pár let porozumíme?