Masarykova univerzita
Fakulta informatiky

Srovnání RAD platforem Seam Forge a Spring Roo

Bakalářská práce

Jan Holman

Brno, jaro 2013

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Jan Holman

Vedoucí práce: Mgr. Marek Grác, Ph.D.

Poděkování

Shrnutí

Klíčová slova

Obsah

1 Úvod

1.1 Cíle práce

1.2 Struktura práce

2 Rapid Application Development

> *Some consider it noble to have a method; others consider it noble not to have a method. Not to have a method is bad; to stop entirely at method is still worse. One should at first observe rules severely, then change them in an intelligent way. The aim of possessing method is to seem finally as if one had no method.*
>
> – The Mustard Seed Garden Manual of Painting

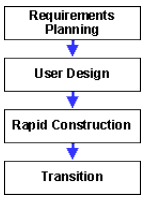
Rapid Application Development, v překladu „rychlý vývoj aplikací“ je metodologie tvorby (mj.) softwaru, která, jak už název napovídá, upřednostňuje rychlost vytváření funkčních prototypů tradičně na úkor použitelnosti, rozsahu implementovaných funkcí a / nebo výkonu. RAD také značně omezuje část plánování ve prospěch samotného vývoje, který probíhá iterativně. Důraz se klade na tvorbu prototypů, na aktivní komunikaci s klientem a jeho participaci na vývoji citace. Metoda byla poprvé popsána v roce 1991 Jamesem Martinem v knize Rapid Application Development jako reakce na tehdejší metody vývoje, které zejména nedokázaly dostatečně pružně reagovat na změny požadavků v průběhu vývoje, a na základě potřeby dodat co nejrychleji fungující systém. [3]

2.1 Rozdíly oproti jiným metodám

Hlavní problém tehdy nejrozšířenější metody tzv. vodopádu je ten, že vývoj systému trvá příliš dlouho. V průběhu vleklého vývoje se mohou klientovy potřeby změnit, takže výsledkem je sice kompletní a technicky dokonalý ale zároveň nepoužitelný systém. Čím déle vývoj trvá, tím vyšší je pravděpodobnost změny v požadavcích, a na tyto změny je třeba pružně reagovat. Oproti běžným agilním metodám se RAD soustředí na vývoj pro uživatele nejzajímavějších částí (?) – vychází z předpokladu, že uživatelé nejčastěji využívají funkce, které jsou pro ně zajímavé, a proto mají tyto části vyšší prioritu. citace RAD také klade důraz na co nejrychlejší naplnění potřeby klienta, spíš než na technickou a technologickou dokonalost. Hlavní oblasti, ze kterých plyne rychlost vývoje pomocí RAD jsou prototypování, iterace a tzv. timeboxing.
Prototypování spočívá v co možná nejrychlejší vytvoření fungujícího proto-

typu (řádově v jednotkách dní), který se zaměřuje pouze na klíčové funkce, a v jeho následném ladění na základě odezvy od klienta a budoucích uživatelů. Nejde pouze o jednorázové prototypy, jako u jiných agilních metod (např.), celý software vzniká iterativně postupným rozšiřováním prvotního prototypu. Prototyp slouží jako důkaz práce pro klienta (například při vývoji pomocí vodopádu klient velmi dlouho nevidí žádný nebo minimální výstup) a také jako referenční bod, ze kterého se vychází při dalším zpřesňování požadavků. Rychlé tvorby prvního prototypu se dosahuje využíváním tzv. CASE odkaz dolů nástrojů, které z formálního zápisu požadavků automaticky generují datový model, funkční databázi a aplikační kód.
Podle pravidla timeboxingu je nutné dodávat výsledky každé iterace včas, a to i za cenu, že se část nestihne a přesune se do dalšího cyklu. Je lepší dodat část zadání včas, než dodat všechno, ale nedodržet časový plán. Pokud by se jednotlivé cykly protahovaly, omezovala by se odezva od klienta a metoda by tím ztrácela podstatnou část výhod z plynoucích z iterativního vývoje.
Během vytváření modelu i celého iteračního procesu jsou také aktivně zapojeni uživatelé/klient, kteří se podílejí na návrhu a schvalují prototypy.

2.2 Fáze vývoje pomocí RAD



Obrázek 2.1: Fáze RAD [1]

1. Plánování požadavků: Definování funkcí, procesů, dat a rozsahu sytému. Výsledkem je seznam entit a diagramů, které definují interakce mezi procesy a datovými elementy. Zachycení požadavků v nástroji tak, aby se daly později automaticky převést na datový model a kód (ne pouze v nestrukturovaném dokumentu).

2. Uživatelský návrh: Workshopy s uživateli/klientem, modelování dat a procesů systému, vytvoření funkčního prototypu kritických částí

systému. Detailní rozvedení požadavků a převedení definovaných entit na datový model, formalizování pravidel, tvorba testovacích plánů, návrh obrazovek uživatelského rozhraní a vazeb mezi nimi. Odhad náročnosti vývoje daného systému, vytvoření prvního omezeného prototypu, který se zaměřuje pouze na (předem definované) klíčové funkce, pomocí CASE nástrojů.

3. Rychlý vývoj: Vlastní tvorba samotného aplikačního systému, tvorba uživatelské podpory a implementace pracovních plánů. Vývoj probíhá v krátkých cyklech - vývoj, testování, upřesnění požadavků, další cyklus (s využitím principu timeboxingu). Převod datového modelu na funkční databázi.

4. Nasazení: Akceptační testování, školení uživatelů, konverze dat, nasazení systému do podniku. [2]

2.3 Výhody

Hlavní výhody plynoucí z RAD jsou rychlost a kvalita. Rychlost je zde dána krátkou dobou mezi dodávkami v jednotlivých cyklech, především díky využívání CASE nástrojů, které ve velmi krátkém čase konvertují požadavky na kód, a principu timeboxingu, tedy striktního dodržování časového plánu. Kvalita se v případě RAD nechápe v tradičním (a asi intuitivnějším) významu, tedy ve smyslu míry, do jaké výsledný systém splňuje zadané požadavky, a míry, do jaké v něm po dodání nedochází k závadám, ale spíš se chápe jako míra, do jaké systém naplňuje potřeby klienta a do jaké má malé náklady na údržbu. Vysoké kvality se v případě RAD dosahuje pomocí aktivního zapojené budoucích uživatelů do celého procesu tvorby systému, především ve fázích analýzy a návrhu. [3]

2.4 Nevýhody

Problémy, které mohou při vývoji pomocí RAD potenciálně vzniknout, jsou především horší škálovatelnost a omezený rozsah vytvořeného systému. Protože při vývoji nejprve vzniká jednoduchý prototyp, který se později iterativně rozvíjí do kompletní aplikace, může výsledek škálovat hůř, než řešení, které je od počátku navržené jako kompletní aplikace. Výsledná aplikace může

také vlivem timeboxingu, tedy upřednostňováním včasného doručení před implementováním všech slíbených funkcí, obsahovat méně funkcí, než aplikace vytvořená tradičním způsobem (například metodou vodopádu).
RAD se samozřejmě nedá použít na všechny typy projektů – například systém řízení letadla vytvořený pomocí RAD by asi moc důvěry nevzbuzoval. Tento přístup je vhodný zejména pro menší projekty, nebo pro projekty, u kterých lze práci rozdělit na více zvládnutelných částí. Stejně tak by měl být poměrně malý i vývojářský tým (ideálně 2-6 lidí) a jeho členové by s tímto typem vývoje měli mít zkušenosti. Nutnou podmínkou jsou dobře definované požadavky a rozsah vytvářené aplikace, malé množství rozhodujících na straně klienta (ideálně 1 člověk) a jejich jasně určení. [3]

2.5 Nástroje podporující metodu RAD

Důležitou součástí vývoje pomocí RAD je využívání kvalitních nástrojů podporujících rychlý vývoj, tedy především CASE nástrojů, které generují aplikační kód. Bohužel ne všechny nástroje, které o sobě prohlašují, že podporují rychlý vývoj, jsou pro metodu RAD skutečně vhodné. Použité nástroje by měly umět strukturovaně zachytit požadavky (UML odkaz dolů nástroje), převést zachycené požadavky na datový model a na jeho základě vygenerovat funkční databázi a velkou část aplikačního kódu.
Základní požadavky na kvalitní nástroj podporující RAD [3] :

- produkuje kód na úrovni enterprise (n-tier) ?
- generuje kompletní prvotní prototyp bez nutnosti přímého psaní kódu (vč. prezentační vrstvy)
- umožňuje plnou kontrolu nad generovaným kódem, například pomocí šablon
- poskytuje flexibilní systém metadat ?
- dá se použít během celého vývoje a zejména nepřepíše kód vývojáře

`http://en.wikipedia.org/wiki/List_of_graphical_user_interface_builders_and_rapid_application_development_tools`

2.6 Spring Roo

2.6.1 Rozšíření

2.7 JBoss Forge

2.7.1 Rozšíření

2.8 Rozdíly

3 Testování UI webových aplikací

3.1 Selenium

3.1.1 Selenium Remote Control

3.1.2 Selenium Grid

3.1.3 Selenium WebDriver

3.2 Arquillian Drone

3.3 Arquillian Graphene 2

4 Vlastní tvorba pluginů

4.1 Plugin pro JBoss Forge

4.1.1 Použité nástroje

4.2 Addon pro Spring Roo

4.2.1 Použité nástroje

5 Závěr

Seznam obrázků

Seznam tabulek

Literatura

[1] *Článek o RAD.* [online]. [2012] [cit. 2013-12-17]. Dostupné z: `http://www.projectmanagement.com/content/processes/11306.cfm`

[2] *Slidy k přednášce o RAD.* [online]. [2012] [cit. 2013-12-17]. Dostupné z: `http://www.ftms.edu.my/pdf/Download/PostgraduateStudent/IMM006%20RAPID%20APPLICATION%20DEVELOPMENT%20-%20Power%20Point%20Slide%20chapter%201.pdf`

[3] *Článek o RAD.* [online]. [2012] [cit. 2013-12-17]. Dostupné z: `http://www.blueink.biz/RapidApplicationDevelopment.aspx`