A FRAMEWORK FOR DYNAMIC UPDATING OF JAVA-BASED APPLICATIONS

Martin Genčúr
Master Degree Programme (2), FIT BUT
E-mail: xgencu00@stud.fit.vutbr.cz

Supervised by: Marek Rychlý
E-mail: rychly@fit.vutbr.cz

ABSTRACT

This paper deals with dynamic software evolution of applications written in Java. A process of software evolution is described which includes an exchange of certain part of the application. A framework for dynamic updating of Java-based applications is introduced.

1. ÚVOD

Moderní společnosti využívají softwarové produkty, které musí spolehlivě fungovat a neustále plnit aktuální požadavky společnosti. Aby software byl schopen takovou funkčnost poskytnout, měl by se dynamicky vyvíjet. Problém softwarové evoluce obvykle vzniká po zavedení první verze softwarového produktu a typicky se týká opravy chyb a přidání nebo změny funkcionality systému. Jistým řešením dynamického vývoje software je právě Rámec pro dynamickou aktualizaci aplikací v jazyce Java. Rámec je schopen aktualizovat objekt v běžící aplikaci a to v případech, kdy není používán, v případě používání v jednovláknové i vícevláknové aplikaci. Jinými slovy, není nutné aplikaci stáhnout z provozu, pokud chceme opravit chybu nebo změnit její chování, jak je u většiny aplikací zvykem. Uživateli tak nevznikají škody spojené se zastavením provozu aplikace.

2. AKTUALIZACE APLIKACÍ V JAZYCE JAVA

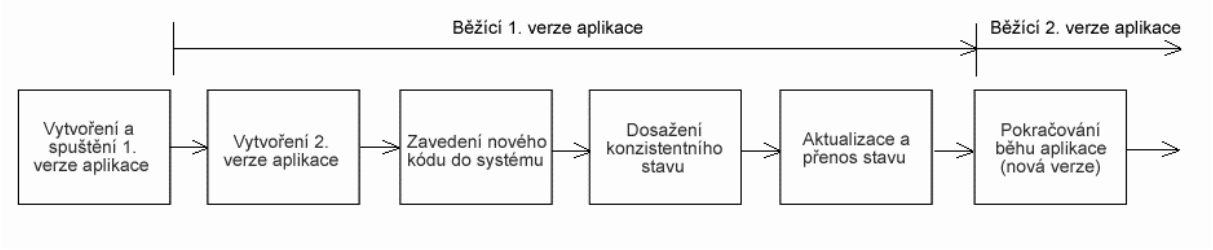
Následující sekce popisují obecnou problematiku spojenou s aktualizací aplikací. Zejména je nastíněno řešení dané problematiky z pohledu rámce pro dynamickou aktualizaci aplikací.

2.1. PROBLEMATIKA AKTUALIZACE SOFTWARE

Proces dynamické evoluce software zahrnuje několik kroků. Obvyklé postupy se mohou lišit napříč programovacími jazyky a platformami. Obvykle lze ovšem proces aktualizace popsat v následujících bodech [1]:

- Vytvoření nové verze aplikace: Tato fáze představuje nezbytnou činnost v procesu aktualizace a sice implementaci nové verze software. Obvykle je k dispozici původní verze zdrojového kódu programu, protože novější verzi vytváří autor původní aplikace.

- Zavedení nového kódu do systému: Některé programovací jazyky s propracovaným mechanismem reflexe, jako např. SmallTalk dovolují snadné provádění změn, včetně definování nových metod programových tříd. Složitější přidávání kódu do běžící aplikace umožňují staticky typované jazyky jako Java nebo C#.
- Dosažení konzistentního stavu aplikace: Aby bylo vůbec možné přenést stav ze starší verze aplikace do nové, musí být dosaženo konzistentního stavu. To znamená buď deaktivaci všech komponent, jichž se změna týká nebo vyčkání na okamžik, kdy se systém v konzistentním stavu nachází.
- Transformace komponent: Tato fáze zahrnuje jednak aktualizaci chování programu, ale také přenos stavu z dřívější verze aplikace do nové. V jazyce Java je chování definováno v metodách tříd a objektů, aktualizace chování tedy znamená modifikaci těchto metod. Složitější je v této fázi ovšem přenos a transformace stavu. Stav je uchováván v instančních a třídních proměnných. Přenos stavu tedy znamená přenos a transformaci těchto hodnot.
- Pokračování běhu aplikace: V tomto okamžiku je nová verze aplikace připravena, spuštěna a pokračuje v činnosti tam, kde starší verze skončila.



Obrázek 1: Fáze životního cyklu aktualizované aplikace.

2.2. ZPŮSOB AKTUALIZACE V JAZYCE JAVA

Java je třídně orientovaný jazyk, což znamená, že jednotkou programu zapouzdřující své chování je třída. Předmětem aktualizace u třídně orientovaných jazyků jsou tedy samotné třídy, potažmo objekty, které jsou z nich vytvářeny.

Důležitým aspektem je fakt, že Java sama o sobě nepodporuje přímou aktualizaci, kdy by původní verze objektu (třídy) byla nahrazena novou verzí. Pouze je možné do systému zavést novou verzi třídy, ovšem mezi původní a novou verzí není žádná spojitost. Docílit schopností dynamické evoluce software lze v Javě v podstatě dvěma způsoby, které ovšem obcházejí rysy virtuálního stroje, což je výkonné jádro Javy. Prvním způsobem je modifikace virtuálního stroje tak, aby dynamické změny podporoval. Nevýhodou je potom nutnost použití modifikované verze virtuálního stroje. Druhým způsobem je vytvoření nějaké nadstavby v podobě knihoven. Výhodou metody je, že nezasahuje přímo do implementace platformy Java a je široce použitelná. Rámec pro dynamickou aktualizaci využívá právě nadstavby v podobě knihoven a dalších pomocných souborů.

2.3. POPIS ŘEŠENÍ RÁMCE

Jak již bylo zmíněno, rámec řeší problém dynamické aktualizace pomocí programové nadstavby. Je to v podstatě sada tříd v jazyce Java, webových formulářů a pomocných souborů ve formátu XML.

Použití rámce v aplikaci umožňující aktualizaci: Pokud vyžaduje aplikace, aby bylo možné aktualizovat třídu, kterou využívá pro implementaci své funkčnosti, musí být tato třída používána specifickým způsobem. Pro vytvoření objektu z dané třídy stačí zavolat funkci rámce, které jsou předány následující parametry: jméno třídy, z níž má být objekt vytvořen a parametry, které by byly předány při vytváření objektu klasickým způsobem. Výsledek funkce je uložen do proměnné, s jejíž pomocí jsou později volány metody objektu normálním způsobem. Proměnná musí být typu interface, podle sémantiky jazyka Java.

Rozhraní pro přístup k aktualizaci: Samotná aktualizace je navržena s využitím webové služby. Tuto službu lze využívat vzdáleně přes internet, přičemž je nutné jí předat potřebné parametry. Takovými parametry jsou zejména adresa aplikace, jejíž objekt se má aktualizovat, název původní verze třídy a nový soubor třídy s příponou .class. Tento nový soubor nahradí v systému předešlou verzi a v nejbližším možném okamžiku jsou objekty aktualizovány tak, aby korespondovaly s touto novou verzí třídy.

Mapování stavu mezi původní a novou verzí objektu: Přenos stavu mezi původní a novou verzí objektu představuje kopírování hodnot datových členů mezi jednotlivými verzemi objektů. Tuto akci dokáže rámec provést automaticky, pokud se jména a vlastnosti datových členů v různých verzích shodují. Pokud tomu tak není, může uživatel, který provádí aktualizaci, dodat vlastní soubor s mapováním datových členů ve formátu XML. Tento soubor je v tom případě zaslán aplikaci zároveň s ostatními parametry popisovanými dříve.

Vylepšení rámce: Implementovaný rámec oproti klasickým aplikacím umožňuje použít aktualizovatelný objekt i ve vícevláknové aplikaci. I v případě, že je objekt vytvořen a předán k dispozici více vláknům, proběhne aktualizace správně. Volání metod objektů a aktualizace jsou synchronizovány pomocí prostředků poskytovaných jazykem Java. Další výhodou rámce je, že vytvářené objekty jsou ukládány do tzv. poolu (rezervy), odkud jsou později k dispozici dalším požadavkům na vytvoření objektu. Tím je značně urychlen běh aplikace, jelikož nemusí být vždy vytvořen nový objekt, což může být časově náročné. Po určité době nečinnosti jsou objekty z poolu automaticky odstraněny.

3. ZÁVĚR

Tento příspěvek poskytl zjednodušený náhled do problematiky aktualizace softwarových aplikací. Zároveň bylo popsáno řešení daného problému z pohledu rámce pro dynamickou aktualizaci aplikací v jazyce Java. Rámec je široce použitelný a to bez nutnosti změny virtuálního stroje Javy. Řešení bylo využito ve webové aplikaci pro řazení obsahu textového souboru. Uživatel měl možnost nahrát na server textový soubor, jehož řádky chce seřadit. Úspěšně byl původní objekt provádějící řazení nahrazen za běhu aplikace za jeho novou verzi a to včetně přenosu stavu objektu.

LITERATURA

[1] Vandewoude, Yves: Dynamically updating component-oriented systems. [Disertační práce], Katholieke Universiteit Leuven, Leuven, 2007, Dokument dostupný na URL http://www.cs.kuleuven.ac.be