Systém pro vývoj distribuovaných aplikací Plaant

Rudolf Pecinovský[1], Vladimír Lahoda[1]

[1]Amaio Technologies Inc., 100 00 Praha 10, Třebohostická 14
rpecinovsky@amaio.com
vlahoda@amaio.com

Abstrakt. Při vývoji distribuovaných podnikových aplikací na zakázku se setkáváme s poměrně typickou sadou požadavků, které se neustále opakují. Systém *Plaant* byl vyvinut jako nástroj pro vývoj a následnou údržbu distribuovaných databázových aplikací. Nástroj, který výrazně snižuje jejich pořizovací i provozní náklady. Současně snižuje požadavky na kvalifikaci a zkušenosti vývojářů, kteří v něm tyto aplikace vyvíjejí. Celý systém je postaven na platformě J2EE a nabízí řadu jedinečných sofistikovaných vlastností a funkcí. Vedle možnosti snadné definice základních formulářů, které systém automaticky upraví pro různé druhy klientů (tlustý klient, webové rozhraní, mobilní telefon, …), nabízí i propracované možnosti řízení koloběhu dokumentů (workflow) a tvorby komplexních sestav určených pro různé cíle (tiskárna, komfortní grafický výstup, webová služba, …).

Klíčová slova: framework, distribuované aplikace, vývoj, J2EE

1 Úvod

Při vývoji distribuovaných podnikových aplikací pro naše zákazníky jsme se setkávali se stále stejnými požadavky a problémy, které byly vždy jen trochu modifikované. Cítili jsme potřebu používat při vývoji těchto aplikací nějaký sofistikovaný systém, který by s těmito typickými požadavky předem počítal a řešil je automaticky, čímž by nám umožnil se soustředit na vlastní logiku aplikace. Protože na trhu žádný takový systém nebyl, rozhodli jsme se jej vyvinout sami. Výsledný vývojový a provozní nástroj jsme nazvali *Plaant*.

2 Základní charakteristika *Plantu* a v něm vytvořených aplikací

Plaant je systém umožňující rychlý vývoj, jednoduché nasazení a snadnou správu robustních datově orientovaných distribuovaných aplikací. Uživatelé aplikací vyvinutých v systému *Plaant* se mohou k těmto aplikacím připojovat prostřednictvím mnoha druhů klientských zařízení od klasického stolního počítače připojeného k místní síti, přes webový terminál až po mobilní telefony programovatelné v Javě (dnes již téměř všechny). Komponentová podstata aplikací vytvořených v systému *Plaant* navíc umožňuje jejich snadnou rozšiřitelnost.

Aplikace vytvořené v systému *Plaant* jsou postavené na platformě J2EE a respektují všeobecně zavedené standardy. To přináší jejich snadnou nasaditelnost na nejrůz-

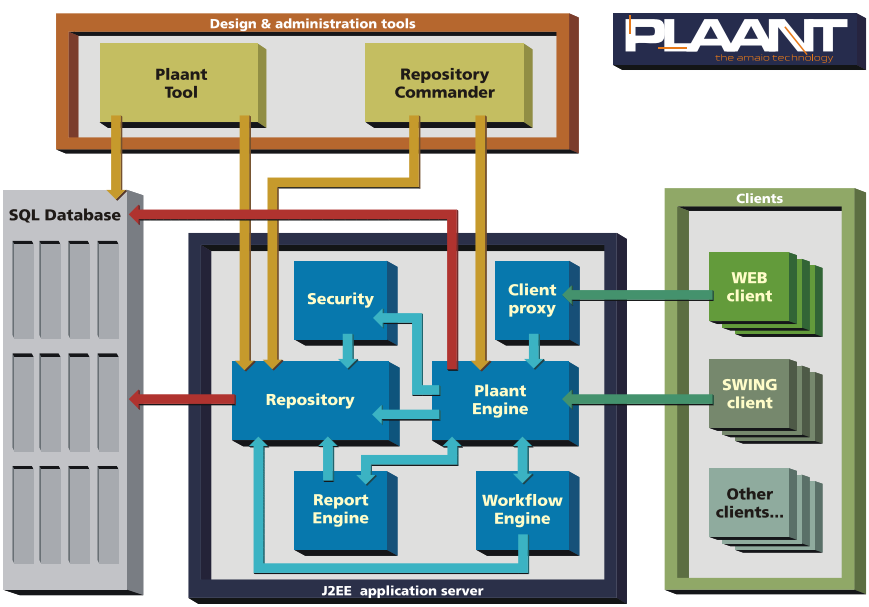
nější počítače a operační systémy a schopnost komunikovat se širokým spektrem používaných databází a aplikačních serverů.

Plaant poskytuje automatizované mechanizmy správy databáze, poštovních služeb a posílání zpráv. Navíc je pro klientské stanice schopen samostatně připravit kompletní grafické uživatelské rozhraní, které již nevyžaduje žádnou další údržbu, a to ani při změnách požadavků. Osvobozuje tak vývojáře od značné části kódování, protože podstatnou část aplikace vytvoří systém sám na základě dodané specifikace jejího požadovaného chování. Tím se veškeré vývojové práce samozřejmě výrazně zrychlí. Zkušenost ukázala, že použitím systému *Plaant* se doba vývoje aplikace a s ní i vývojové a udržovací náklady sníží zhruba na polovinu (40 až 60 %).

Klienti vytvoření systémem *Plaant* nevyžadují, nezávisle na svém typu, žádnou dodatečnou údržbu. Po prvotním nastavení klientského zavaděče aplikace jsou všechny aktualizace realizovány automaticky.

3 Části systému *Plaant*

Systém *Plaant* je tvořen několika relativně nezávislými částmi, z nichž některé se uplatní při vývoji nových aplikací, některé při jejich provozu a další v obou etapách života aplikace.



Obr. 1: Architektura systému *Plaant*

3.1 Prostředky pro návrh a správu aplikace

Plaant Tool

Plaant Tool je samostatný program, který je základním integrovaným vývojovým prostředím používaným při návrhu aplikace. Výrazně usnadňuje návrh, vytvoření a nasazení (deployment) aplikace. Poskytuje prostředky pro návrh základní struktury databáze (detaily struktury dotvoří systém sám) spolu s prostředky pro návrh uživatelského rozhraní a automatizuje vytvoření potřebné struktury tabulek a následně nasazení aplikace na aplikační server.

Způsob návrhu uživatelského rozhraní, který *Plaant Tools* využívá, není sice klasický WYSIWYG, nicméně umožňuje velmi snadno navrhnout i relativně komplikované formuláře. V kterémkoliv okamžiku pak na požádání zobrazí aktuální podobu navrhovaného formuláře ve swingovém či webovém rozhraní, takže návrhář si může ihned zkontrolovat, nakolik jeho návrh odpovídá požadavkům zadavatele.

Práce s programem *Plaant Tool* nevyžaduje žádné programátorské znalosti a zkušenosti. Při vhodném zaškolení může s jeho pomocí vyvíjet základních kostru budoucích aplikací i zkušenější uživatel.

Plaant Commander

Plaant Commander je také samostatný program a slouží k přípravě vytvořené a nasazené aplikace k jejímu prvnímu spuštění. V dalších etapách jejího života je to pak základní pracovní nástroj jejího správce. Jeho GIU je odvozeno od oblíbených správců souborů à la *Norton Commander*, takže uživatelům nečiní žádné problémy.

3.2 Jádro aplikací *Plaant*

Jádro plaantových aplikací je tvořeno několika spolupracujícími komponentami. Využívají je nejenom vlastní aplikace, ale i výše zmíněné nástroje *Plaant Tool* a *Plaant Commander*.

Komponenty systému *Plaant* vyhovují specifikaci EJB 1.1, takže mohou být provozovány i na starších verzích aplikačních serverů. Organizace, které jeho aplikace nasazují, proto nejsou nuceny vyměnit používané systémy, což často výrazně šetří instalační náklady.

3.3 Centralizovaná správa datových struktur

Všechny informace o systémových datových strukturách aplikace, jejích komponentách a uživatelském rozhraní jsou uloženy v datovém úložišti (repository). Při spuštění si aplikace tyto informace přečte a podle nich se nastaví. Stejně tak i klient obdrží při přihlášení k aplikaci informace o struktuře dat a uživatelského rozhraní a podle nich se pak nastaví.

Tato koncepce výrazně urychluje vývoj aplikace a zároveň chrání systém před nekorektními úpravami, při nichž vývojáři zapomenou na některé důležitě aspekty či vazby.

Při každé změně uložených informací *Plaant* zkontroluje jejich korektnost a v případě odhalení jakékoliv nekorektnosti ji odmítne zanést do úložiště a upozorní vývojáře na to, aby ji opravil.

Na druhou stranu tato koncepce umožňuje kdykoliv snadno, rychle a s minimálními náklady změnit některé vlastnosti aplikace či upravit funkčnost jednotlivých komponent.

4 Vlastnosti vytvořených aplikací

Křižovatka
Po spuštění aplikace se otevře aplikační okno s panelem označovaným jako *Křižovatka*. Na tomto panelu je několik karet přiřazených skupinám tzv. agend (pojem bude vysvětlen dále). Na každé kartě pak uživatel nalezne tlačítka zastupující dostupné agendy. Stiskem tlačítka se pak přímo přesune do panelu dané agendy.

Tabulky a formuláře
Panel agendy má opět několik karet. Jako první je vždy uvedena karta tabulky, což je základní zobrazení, které se nastaví při přechodu do okna agendy. V tabulce je možno se pohybovat po záznamech v agendě, vyhledávat je, řadit a filtrovat.

Výběr polí a řazení záznamů
Na kartě tabulky si může uživatel kdykoliv vybrat, která z polí příslušné agendy, resp. z agend, na jejichž záznamy dané agendy odkazují (resp. z agend odkazovaných z odkazovaných agend atd.) budou v tabulce zobrazena a v jakém pořadí. Současně může zadat, zda budou záznamy nějak seřazeny, přičemž je může řadit i podle několika polí současně (např. příjmení – křestní jméno – rodné číslo).

Filtry
Plaant nabízí bohaté a propracované možnosti filtrování zobrazovaných záznamů. Filtrovat lze přitom nejenom podle hodnot v polích (zobrazovaných i nezobrazovaných), ale podle libovolně složitých logických výrazů, v nichž porovnání těchto hodnot vystupují. Při specifikaci rozhodovacích pravidel se přitom uživatel nemusí omezovat pouze na hodnoty polí dané agendy, ale může se odkázat na libovolné pole kterékoliv z agend, na jejíž záznamy se záznamy dané agendy odkazují, na pole agend odkazovaných z odkazovaných agend atd. To vše bez nutnosti explicitního uvádění vazeb mezi jednotlivými tabulkami. Troufáme si tvrdit, že prostředky, které takto koncipované filtry poskytují, jsou daleko mocnější a přitom uživatelsky pochopitelnější než dotazovací mechanizmus programu Access, jenž je na mnoha místech vydáván za vzor uživatelské přívětivosti.

Sestavy
Aplikace jsou připraveny i na situace, kdy uživatel potřebuje „vyjet“ nějakou sestavu, která reaguje na okamžitou situaci a v návrhu aplikace se s ní nepočítalo. Pro definici sestav je k dispozici nástroj, který nabízí obdobné možnosti, jaké mají uživatelé pro definici zobrazovaných a třídících polí a k definici filtrů.

Pohledy
Souhrn nastavení zobrazovaných a třídících polí, filtru a sestavy je chápán jako pohled na agendu. Uživatel může definovat řadu různých pohledů a každý si uložit pro případ, že se situace či úloha, pro níž daný pohled definoval (např. účetní uzávěrka, vyhledání faktur po splatnosti, přehled nejdůležitějších zákazníků atd.), bude v budoucnu opakovat.

5 Jak se v systému *Plaant* vyvíjí aplikace

Plaant je komplexní systém. Je to zároveň sada nástrojů pro vytvoření aplikace a zároveň prostředí, v němž na serveru aplikace běží a které komunikuje s koncovými uživateli prostřednictvím klientského GUI a umožňuje administrátorovi správu celé aplikace.

5.1 Analytická příprava – návrh agend

Při analýze úlohy je třeba navrhnout databázovou strukturu budoucí aplikace a funkce, které bude aplikace plnit. Typický objekt databázové aplikace modeluje nějakou datovou entitu – tvoříme-li např. databázi pro knihovnu, budeme (mimo jiné) modelovat jednotlivé knihy, nakladatele, autory, čtenáře atd.; tvoříme-li databázi pro účtárnu, budeme modelovat účty, faktury, zákazníky, dodavatele, položky faktur atd.

Kolem každé modelované entity vzniká celá agenda činností, které jsou s ní spojeny. Tuto agendu má v *Plaant* aplikacích na starosti samostatná komponenta, kterou budeme označovat termínem *Agenda* a která představuje modelovanou entitu jako celek s jejím funkčním i datovým obsahem.

Data
Agendy bývají v relační databázi většinou reprezentovány jako tabulky, ale není to pravidlo. Agendy jsou totiž obecnější. Na jednu stranu mohou existovat tzv. agendy bez dat, které nemají v databázi žádnou přímou reprezentaci a jsou realizovány čistě programově, na druhou stranu však mohou existovat také agendy, které jsou v relační databázi reprezentovány celou skupinou tabulek.

Agendy mohou vytvářet stromy dědičnosti, v nichž dceřiné agendy přebírají do svých záznamů všechna pole svých rodičovských agend. Možnost využití mechanizmu dědičnosti výrazně zjednodušuje analýzu a realizaci některých typických zákaznických požadavků.

Jak bývá v relačních databázích zvykem, agendy mohou obsahovat odkazy na jiné agendy, přičemž *Plaant* rozeznává dva druhy odkazů: jednoduchý odkaz směřující na konkrétní záznam a tabulkový odkaz odkazující na celou množinu záznamů. Při vytváření odkazů typu 1:N přitom nebývá nutno vytvářet explicitně příslušnou mezitabulku, protože ji *Plaant* dokáže vytvořit a spravovat sám.

Agendy mohou obsahovat nejenom odkazy na jiné agendy, ale mohou také obsahovat jiné agendy jako své komponenty. To ulehčuje analýzu i realizaci častých úloh,

které je tak možno předem připravit jako samostatné komponenty a poté je do vytvářené aplikace pouze vložit.

Formuláře
Součástí definice agendy je nejenom návrh jejích polí, ale také návrh formulářů, prostřednictvím nichž budou uživatelé přistupovat k jejím datům. Jak jsme již naznačili, tyto formuláře budou použity pro všechna rozhraní, která bude daná aplikace podporovat, a proto je vhodné na tuto skutečnost při jejich návrhu myslet (např. nevytvářet příliš rozměrné formuláře, počítáme-li s častým použitím aplikace na různých PDA a dalších zařízeních s menší obrazovkou).

Stavové diagramy
U některých aplikací se množina operací, které je třeba provádět, resp. které se smí provádět s daty v záznamu, závislá na stavu, v němž se daný záznam nachází. Takovéto závislosti bývá výhodné popsat stavovým diagramem, v němž můžeme k jednotlivým stavům přiřadit jak množinu přípustných, resp. povinných operací, tak akce, které se automaticky spustí při přechodu mezi zadanými stavy.

Výstupní sestavy
Prakticky všechny aplikace musí být schopny generovat nejrůznější sestavy. *Plaant* v tomto směru vychází zákazníkům vstříc a na požadované formátování a obsah výstupních sestav neklade prakticky žádná omezení.

Speciální funkce
Součástí návrhu aplikace je také zadání speciálních funkcí, které realizují některé nestandardní činnosti. Tyto funkce jsou většinou následně implementovány buď jako spouštěče (triggery) nebo jako průvodci, kteří při vyvolání dané funkce pomohou uživateli postupně specifikovat jeho konkrétní požadavky.

Pro tvorbu případného uživatelského rozhraní těchto speciálních funkcí má návrhář aplikace k dispozici stejné nástroje jako pro definici základního chování agend. Vlastní výkonný kód funkcí je však třeba naprogramovat klasickými prostředky prostřednictvím tříd postavených nad API sytému *Plaant*.

Přístupová práva
Poslední, avšak velmi důležitou součástí zadání je specifikace jednotlivých uživatelských rolí a práv, která jsou s těmito rolemi spojena. Možnosti systému *Plaant* jsou v této oblasti opět nesmírně bohaté. Na rozdíl od běžných standardů má jeho přístup k bezpečnosti nízkou granularitu, takže umožňuje daleko jemnější nastavení nejrůznějších bezpečnostních hledisek.

Bezpečnostní koncepce aplikací vyvíjených pomocí systému *Plaant* není postavena na uživatelích, ale na rolích. Každý uživatel může mít v systému řadu rolí a jeho možnosti přístupu k informacím jsou ovlivněny jeho aktuální rolí.

Tato koncepce umožňuje snadné a rychlé nastavení i velmi komplikovaných bezpečnostních pravidel a především pak jejich rychlou operativní změnu. Změní-li se např. pozice nějakého uživatele v hierarchii společnosti, stačí mu pouze přiřadit role odpovídající jeho nové pozici a tímto jednoduchým přiřazením se uživateli v okamžiku přiřadí celý komplexní balík práv a omezení odpovídající jeho nové pozici.

5.2 Vytvoření kostry aplikace v *Plaant Tools*

Datová struktura

Po analýze nastupuje návrh kostry aplikace. Návrhář aplikace, který nemusí být programátor, navrhne za pomoci programu *Plaant Tools* základní objektové schéma, tj. definuje, které agendy bude aplikace obsahovat a jaké údaje v nich budou uloženy. Součásti zadávaných informací je i to, zda se má vyžadovat zadání hodnoty, v jakém rozsahu hodnot se smí zadávané hodnoty pohybovat a případně i vzor (šablona) pro vstup i zobrazování dat.

Při té příležitosti vývojář také definuje strom případných dědičností jednotlivých agend a vložení obsahu jedné agendy do druhé jako její komponenty. Jak jsme již řekli, návrhář se při tomto návrhu nemusí zabývat definicí nějakých pomocných tabulek, protože ve většině případů mu stačí pouze odsouhlasit, že zadání vyhovují ty tabulky, které pro takovéto účely definuje *Plaant* automaticky. Má-li však návrhář nějaké nestandardní požadavky, může zadáním příslušných parametrů tvorbu těchto tabulek samozřejmě ovlivnit.

Formuláře

Současně s datovým obsahem agend navrhne vývojář v této etapě i podobu budoucích formulářů. Tuto podobu sice nedefinuje pomocí nějakého WYSIWYG návrháře, ale prostřednictvím nastavení různých parametrů. V každém okamžiku však má možnost zkontrolovat, jak bude navržený formulář vypadat jak v desktopové, tak ve webové verzi GUI.

WYSIWYG návrhář není v systému použit především proto, že se s jeho pomocí špatně nastavuje vzhled formuláře pro několika výrazně odlišných platforem současně (např. pro desktopovou aplikaci, pro webovou aplikaci a pro PDA). Byla proto dána přednost koncepci, při níž tvůrce nedefinuje přesnou konečnou podobu GUI, ale zadává pouze parametry definující sdružování zobrazovaných objektů do větších skupin a vzájemnou relativní pozici těchto objektů a skupin.

Součástí definice formuláře je jak specifikace druhu použitého prvku (vstupní pole, rozbalovací seznam, přepínač, skupina odkazů, …), tak popis rozmístění a rozměrů prvků zobrazujících jednotlivé datové položky, tak sada doplňujících informací, které specifikují, zda bude zobrazovaná hodnota zadávána uživatelem nebo počítána systémem, zda bude požadováno její povinné zadání, kontrola povolených hodnot, šablona pro zadávané hodnoty a některé další informace.

Lokalizace

Součástí základního návrhu je i lokalizace používaných pojmů pro jednotlivé jazykové mutace. Aplikace pak při přihlášení každého klienta připraví svoji lokalizovanou mutaci podle lokality nastavené v klientském systému, případně zadané ve spouštěcím příkazu.

Výstupy *Plaant Tools*

Na základě informací zadaných ve výše popsaných fázích vytvoří *Plaant Tools* sadu XML souborů, do nichž vloží informace o jednotlivých funkčních segmentech aplikace, odpovídajících datových strukturách a klientském uživatelském rozhraní. Tyto

soubory aplikace při svém spouštění načte a podle získaných informací vše potřebné nastaví.

Současně *Plaant Tools* připraví skripty pro použitý RDBMS (Relational Database Management System). Tyto skripty zabezpečí vytvoření potřebných tabulek v databázi, a to jak tabulek s informacemi potřebnými pro vlastní činnost aplikace, tak tabulek pro uložení uživatelských dat.

Nejedná-li se o počáteční návrh, ale modifikaci nějakého staršího návrhu, generuje *Plaant Tools* skripty, které pouze modifikují stávající databázi, aniž by přitom došlo ke ztrátě dat.

5.3 Další „neprogramové“ části návrhu

Stavové diagramy

Součástí návrhu aplikace mohou být i stavové diagramy specifikující stavy, kterými mohou procházet záznamy jednotlivých agend, možné operace s daty v jednotlivých stavech záznamu i operace, které se spouští při přechodu záznamu z jednoho stavu do druhého (např. automatické odeslání mailu po vypršení doby splatnosti faktury).

Stavový diagram může vývojář navrhnout v jakémkoliv nástroji pro kreslení UML diagramů, který je schopen ukládat diagramy ve standardizovaném formátu XMI. Tyto diagramy pak aplikace při svém spouštění načte a podle získaných informací nastaví potřebné vnitřní parametry.

Generátor sestav *XANDY*

Většina aplikací se neomezuje pouze na komunikaci s uživatelem prostřednictvím obrazovky, ale součástí zadání je i schopnost generace nejrůznějších sestav. Poměrně propracovaný, nicméně jednoduše ovladatelný generátor sestav je součástí každé aplikace a může jej použít kterýkoliv uživatel, a to zejména při vytváření některých speciálních (často jednoúčelových) sestav, případně pro export tabulkových dat. Pro opravdu náročné sestavy, u nichž je požadováno propracované formátování, se však nehodí. Takovéto sestavy je třeba navrhnout ve fázi návrhu aplikace.

Pro návrh sestav a dalších složitě formátovaných dokumentů byl vyvinut program *XANDY*, což je specializovaný XML editor podporující tvorbu formátovaných dokumentů ve WYSIWYG režimu. Výstupem tohoto programu je opět sada XML dokumentů, které si aplikace při svém spuštění načte.

5.4 Programové doladění

Pro všechny doposud popsané činnosti nebylo potřeba, aby měl vývojář nějaké programátorské znalosti a zkušenosti. Nicméně prakticky každá aplikace vyžaduje jisté schopnosti a funkce, které se pomocí výše popsaných nástrojů zabezpečit nedají. Tyto schopnosti bychom mohli rozdělit do dvou skupin: na spouštěče (triggery), které jsou automaticky spouštěny při nějaké události (vytvoření záznamu, uložení záznamu, …), a speciální funkce, které jsou přístupné uživateli. Všechny se vyvíjejí za pomoci standardních nástrojů pro vývoj aplikací v jazyce Java a pro jejich vývoj je potřeba, aby měl vývojář jisté programátorské zkušenosti a znal API systému *Plaant*.

Spouštěče (triggery)
Pro každou agendu je možno definovat její vlastní třídu, která obsahuje předem danou sadu metod automaticky spouštěných při předem známých událostech, jakými jsou např. otevření nového prázdného záznamu (zde se většinou předvyplní některá pole), uložení nově vytvořeného záznamu (v tuto chvíli se provádí řada nestandardních kontrol, tj. těch, které není možno nastavit v programu *Plaant Tools*), otevření existujícího záznamu, uložení modifikovaného záznamu, změna hodnoty sledovaného pole, nastavení speciálního filtru, atd. atd.

Speciální funkce
Speciální funkce realizují buď operace, které může uživatel přímo spouštět stiskem příslušného tlačítka v aplikačním okně (každá uživatelem spustitelná speciální funkce má v okně vlastní tlačítko), nebo operace automaticky spouštěné např. při přechodu mezi jednotlivými stavy dokumentu. Uživatelem spouštěné operace jsou většinou koncipovány jako průvodci, kteří uživatele nejprve požádají o přesnější specifikaci jeho požadavků a na závěr provedou požadovanou operaci.

5.5 Příprava nasazení aplikace

Před nasazením aplikace je třeba definovat role, specifikovat práva a omezení kladená na jednotlivé role, připravit uživatelská konta, přiřadit uživatelům jejich role a nastavit výchozí podobu uživatelského rozhraní pro jednotlivé skupiny uživatelů. K tomu slouží program *Plaant Commander*, jehož uživatelské rozhraní je podobné klasickým souborovým správcům odvozeným od programu *Norton Commander*, a to včetně nejznámějších klávesových zkratek. Jeho ovládání je proto pro většinu správců naprosto intuitivní a nemívají s ním problémy.

6 Závěr

Nasazení systému *Plaant* přináší řadu konkurenčních výhod. Jedny z nejvýraznějších jsou výhody při vývoji. Vývoj rozsáhlých podnikových aplikací je totiž vysoce odborně i časově náročný a tím také velmi drahý. Použití systému *Plaant* umožní mnohé z této náročnosti výrazně eliminovat. Zkušenosti získané při použití tohoto systému u několika velkých zákazníků ukazují, že jeho nejdůležitější výhody, které se projeví při vývoji systému, jsou:

- Doba od počátku vývoje do výsledného zprovoznění aplikace nebo jejího uvedení na trh se zkrátí o 30 %.
- Počáteční vývojové náklady se sníží o 25 až 40 %.
- Použití systému dokáže výrazně snížit požadavky na kvalifikaci a předchozí zkušenost vývojářů.
- Vývojové náklady během celého životního cyklu aplikace se sníží v průměru o 40 až 60 %.
- Systém *Plaant* umožňuje snadnou a především rychlou úpravu uživatelského rozhraní podle měnících se požadavků uživatelů.

- Systém *Plaant* umožňuje vývojářům vyvíjet uživatelské rozhraní pouze jednou. Systém pak sám realizuje potřebné modifikace pro jednotlivé typy klientů.

Aplikace vyvinuté pod systémem *Plaant* jsou nejenom vyvinuté rychle a levně, ale nabízejí zároveň široké možnost při následné správě systému a vysokou variabilitu při nastavování nejrůznějších okrajových podmínek a požadavků.

Koncovým uživatelům pak nabízí rozsáhlé možnosti individuálního přizpůsobení aplikace jejich konkrétním okamžitým potřebám a možnost kdykoliv snadno upravit výchozí uživatelské nastavení.

Reference

1. Sun, Java 2 Platform, Enterprise Edition (J2EE), *http://java.sun.com/j2ee*.

Annotation

In the custom development of distributed enterprise applications we encounter typical set of repeating requirements. Therefore we developed system *Plaant* as the tool for development and following management of distributed database applications. The tool, which significantly decreases their total cost of ownership. At the same time it decreases the demands for developers qualification and experience. The system is built on the J2EE platform and offers the large set of unique and sophisticated features and functions. Besides the possibility of simple definition of the basic forms, which are automatically adapted for different types of clients (thin client, web interface, mobile phone, …) it also offers sophisticated functions for control of the documents workflow and for preparing of the complex reports for different targets (printer, comfortable graphical output, web service, …).