

Využití CASE ve vývojářské firmě

CASE study + trendy a obecné závěry

27.12.2009

Semestrální práce k předmětu 4IT450 CASE

Vlastimil Vagner, Václav Slavětínský, Andrei Nazarov, Denis Havránek, Martin David

Obsah

1	Úvod	6
1.1	Cíle práce.....	6
2	Vývoj a CASE	2
2.1	Přínosy využití CASE nástrojů při vývoji SW	2
2.2	Vlastnosti CASE.....	2
2.3	Podporované procesy vývoje.....	3
2.4	Komponenty CASE nástrojů	6
2.5	Metodiky vývoje.....	7
2.6	Používané typy diagramů	8
2.6.1	Skupina diagramů UML	8
2.6.2	Diagram podnikových procesů.....	9
2.6.3	Entity Relationship diagram	9
2.6.4	Diagram datových toků	9
2.6.5	Diagramy pro modelování XML	9
2.6.6	Diagramy pro CABE, řízení projektů a další.....	9
2.7	Přehled dostupných SW nástrojů.....	10
2.7.1	PowerDesigner 12.5.....	10
2.7.2	Oracle Designer 10g	11
2.7.3	Enterprise Architect 7.5	13
2.7.4	Select Architect 7.0.....	14
2.7.5	Rational Software Architect V7.5.....	15
2.7.6	Další nástroje	16
2.7.7	NetBeans IDE 6.8	16
2.7.8	Eclipse 3.5.1 (Galileo).....	18
2.7.9	Visual Studio 2010 Beta 2 Ultimate	19
3	Výběr vhodného nástroje CASE.....	20
3.1	Cíle společnosti zabývající se vývojem softwaru	20
3.2	Technologie	20
3.3	Metriky využitelnosti CASE nástrojů.....	21
3.3.1	O metrikách	21
3.3.2	Funkcionalita	22
3.3.3	Použitelnost	22
3.4	Problémy začínajících firem v oblasti CASE.....	26

3.5	Perspektivy	26
1.1.1	Eclipse Modeling Framework (EMF)	27
4	Průzkum trhu o využívání CASE nástrojů	29
4.1	Cíle průzkumu	29
4.2	Analýza výsledků	29
4.2.1	Trend používání CASE	29
4.2.2	Používané nástroje a technologie	30
4.2.3	Přínosy a výhody	30
4.2.4	Nevýhody a omezení	31
4.2.5	Očekávaná zlepšení	31
4.3	Zhodnocení.....	31
5	Závěr	32

Seznam obrázků

Obrázek 1: Pokrytí fází životního cyklu aplikace druhu CASE	4
Obrázek 2: Netbeans IDE 6.5	16
Obrázek 3: Schéma Eclipse Modeling Framework (EMF)	27
Obrázek 4: Schéma - Ecore EMF Meta model.....	27
Obrázek 5: Jednoduchý příklad EMF Ecore Meta modelu	28

Seznam tabulek

Tabulka 1: Přehled vlastností PowerDesigner 12.5.....	Chyba! Záložka není definována.
Tabulka 2: Přehled vlastností Oracle Designer 10g	Chyba! Záložka není definována.
Tabulka 3: Přehled vlastností Enterprise Architect 7.5	Chyba! Záložka není definována.
Tabulka 4: Přehled vlastností Select Architect 7.0.....	Chyba! Záložka není definována.
Tabulka 5: Přehled vlastností Rational Software Architect V7.5.....	Chyba! Záložka není definována.
Tabulka 6: Přehled vlastností NetBeans IDE 6.8	17
Tabulka 7: Přehled vlastností Eclipse 3.5.1 (Galileo)	18
Tabulka 8: Přehled vlastností Visual Studio 2010 Beta 2 Ultimate	19
Tabulka 6: Metriky použitelnosti	25

1 Úvod

Softwarové inženýrství představuje disciplínu kombinující prvky vývoje softwaru a managementu, jejímž hlavním cílem je zavedení a řízení pravidel pro vývoj software. Dnes, stejně jako ve více současných oborech, existuje možnost podpory pomocí moderní technologie a výkonných metodologií řízení. U softwarového inženýrství je za tímto účelem zaveden oficiální pojem CASE.

Zkratka CASE je označením pro Computer Aided Software Engineering (v českém znění tedy „Počítačem podporované softwarové inženýrství“).[2] Jeho význam spočívá v interaktivní podpoře vývoje softwaru za pomoci počítačových technologií. Obecně CASE představuje vlastnost splňující podmínky pro zvýšení efektivity vývoje software. Programová realizace vlastností CASE je označována jako CASE nástroj. Právě těmito pojmy a záležitostmi s nimi souvisejícími se zabývá tato práce.

1.1 Cíle práce

Cílem práce je přiblížení pojmu CASE, jako nástroje pro podporu vývoje softwaru ve vývojářských společnostech.

Práce je rozdělena do tří hlavních částí, které se zabírají:

1. Vývojem CASE nástrojů, podporovanými metodikami a přínosy plynoucí z jejich použití. Detailně jsou pak popsány podporované typy diagramů pro modelování v CASE nástrojích a charakteristiky vybraných CASE produktů.
2. Analýzou ankety, která měla za cíl získat reálný a aktuální pohled na dnes používané CASE nástroje ve vývojářských firmách působících na českém trhu a trendy s nimi spojenými.
3. Obecným pohledem na vývojářské společnosti, jejich konkrétní problémy na současném trhu a nároky, které jsou kladeny na dnešní CASE nástroje.

2 Vývoj a CASE

2.1 Přínosy využití CASE nástrojů při vývoji SW

Jedna z hlavním otázek, kterou si většina čtenářů položí, se týká přínosů, které jim může použití CASE nástrojů při vývoji zajistit.

Mezi nejvýznamnější přínosy pro vývojářskou práci patří:

- Vyšší produktivita práce
- Nižší chybovost
- Snadnější údržba a další vývoj výsledného produktu
- Kvalitnější dokumentace
- Umožnění větší participace uživatelů na vývoji produktu

Existuje mnohem více přínosů, nýbrž jsou odvislé od možností konkrétního nástroje a způsobu jeho použití. Především je nutné, aby člověk využívající těchto nástrojů měl představu, čeho hodlá ve své práci dosáhnout. Toto představuje důležitou metriku použitelnosti CASE nástrojů a určuje jejich samotnou hodnotu. [3]

2.2 Vlastnosti CASE

CASE nástroje představují efektivní pomocné prostředky pro vývoj software. Ačkoli některé nástroje umožňují i generování výsledného kódu ve zvoleném programovacím jazyce, přesto je k samotnému vývoji potřeba vlastnit určité know-how (např. znalost programovacího jazyka či platformy, znalost vývojových metodik a podpůrných technologií, atd.). CASE nástroj však může poskytnout určitý řád a pomoci strukturovat problémy spojené s vývojem aplikace či řízením zdrojů. Většina CASE nástrojů splňuje (přínejmenším by měla) určitou množinu hlavních dovedností, které jsou pro CASE typické.

Mezi hlavní dovednosti CASE nástrojů patří:

- modelování informačních systému pomocí vybraných diagramů (kvůli lepší srozumitelnosti na úrovni člověka)
- generování zdrojového kódu z modelu (usnadňuje vývoj např. rychlým automatickým vytvořením předpřipravených komponent aplikace)
- zpětné vytvoření modelu podle existujícího zdrojového kódu (nebo-li reverzní inženýrství)
- synchronizace modelu a zdrojového kódu
- vytváření dokumentace z modelu

Existuje mnohem více typických vlastností, které CASE nástroje podporují a které jsou neustále z konkurenčních důvodů zdokonalovány. Mimo jiné se objevují také vlastnosti, které jsou typické pro konkrétní nástroje (např. specifické typy diagramů, transformace mezi modely pomocí SVN repozitáře, atd.). K těmto vlastnostem se dostaneme dále v průběhu této práce.

2.3 Podporované procesy vývoje

Na vývoj software je potřeba pohlížet ve více úrovních, než pouze z hlediska vlastní implementace. Takové pohled pokrývá mimo jiné i samotnou přípravu a sběr podkladů pro projekt. Obecně lze klasifikovat vývoj software do 5 hlavních vývojových stádií, kterými jsou:[1]

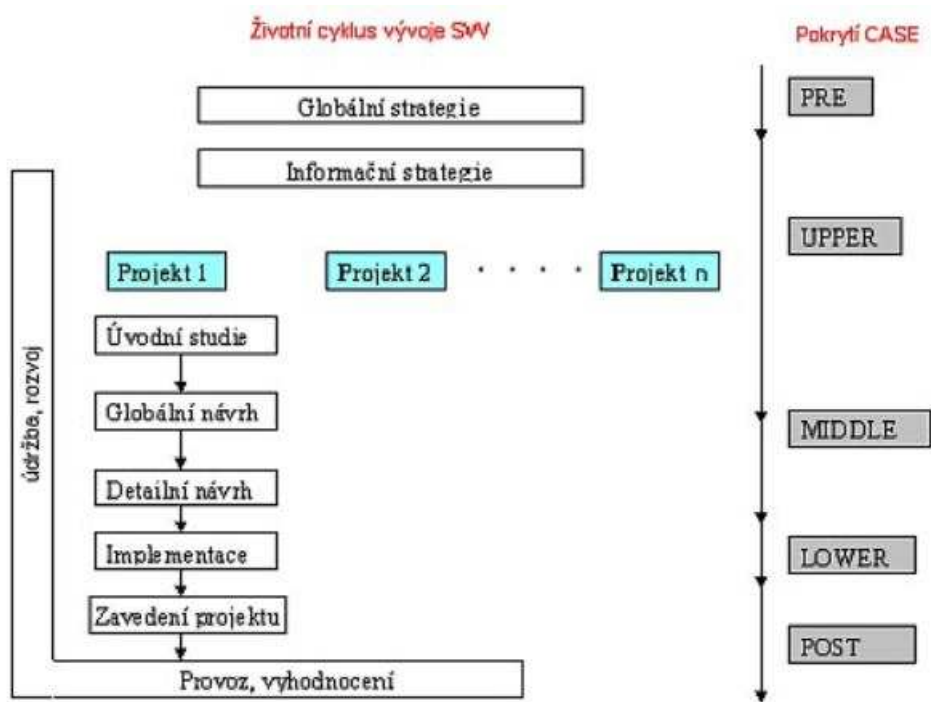
1. **Pre CASE** (podporuje činnosti předcházející vývoji IS – globální strategie)
2. **Upper CASE** (podporuje tvorbu informační strategie a fázi analýzy)
3. **Middle CASE** (podporuje tvorbu globálního a detailního návrhu IS)
4. **Lower CASE** (podporuje fázi implementace)
5. **Post CASE** (podporuje fázi uvedení IS do provozu, provoz, údržbu, reengineering)

Toto dělení bývá označováno jako 5 fází životního cyklu softwarového projektu. Zkráceně se používají někdy pouze 2 nejdůležitější fáze, kterými jsou: [1]

1. **Upper CASE** (podporuje fáze analýzy a návrhu – např. nástroje pro tvorbu diagramu, generování reportu a formulářů a analytické nástroje)
2. **Lower CASE** (podporuje fáze implementace, testování a řízení konfigurací)

Z tohoto pohledu lze zároveň kategorizovat i CASE nástroje, na jehož základě lze přesněji specifikovat, pro které z těchto fází má CASE nástroj podporu (tedy které tyto fáze v něm lze modelovat či realizovat). [1]

Na následujícím obrázku je názorně graficky představen postup vývoje software (včetně paralelně běžících projektů), kategorizovaný do uvedených 5 fází životního cyklu softwarového projektu:



Obrázek 1: Pokrytí fází životního cyklu aplikace druhy CASE¹

Tyto fáze lze přesněji definovat následujícím způsobem: [1]

Pre CASE – podporují tvorbu globální strategie.

Upper CASE – podporují plánování, specifikaci požadavků, modelování organizace podniku a globální analýzu IS. Hlavním úkolem nástroje je analýza organizace, zachycení procesů v organizaci, definice klíčových informačních toků a dokumentace zjištěných požadavků.

Z těchto údajů je jasné použití při specifikaci cílů, počáteční specifikaci požadavků a řízení projektů. Použité nástroje mohou být DFD (Data Flow Diagram) a ERD (Entity Relationship Diagram) bez podrobných atributů, prostředky pro řízení projektů a sledování ekonomických ukazatelů, popis základních vlastností systému prostředky OO modelování.

Middle CASE – podporují podrobnou specifikaci požadavků a vlastní návrh systému. Tato třída CASE nástrojů je nejuspěšnější. Používají se pro podrobnou specifikaci požadavků, návrh systému, dokumentaci a vizualizaci systému. Použité metody a nástroje jsou DFD včetně podrobného popisu procesů, datových úložišť, podrobné ERD, pro OOAN – diagramy tříd, instancí, přechodové diagramy apod.

Dále CASE nástroje této kategorie obsahují systém správy dokumentů a konfigurace, systém pro vyhodnocování metrik, vývoj prototypů, návrh rozhraní. Mohou obsahovat také generátory obrazkových formulářů a tiskových sestav a také generátory (kostry) definic dat. Tento druh CASE je jádrem komerčně dodávaných CASE systémů.

Lower CASE – obsahují nástroje pro podporu kódování, testování, údržby a reverzního inženýrství. Integrované jsou nástroje, jako jsou generátory kódu (mohou generovat jen kostru

¹ <http://www.dbsvet.cz/grafika/cas01.jpg>

nebo až 75 procent výsledného kódu, kde programátor doplňuje většinou jen detaily). Dále pak jde o prostředky pro reverse engineering (rekonstrukce dokumentace a modelů z existujícího SW), prostředky pro sledování a vyhodnocení metrik, prostředky plánování a zjištění kvality SW (sběr informací o průběhu testování, vyhodnocení výsledků testů, řízení testování), pro správu konfigurace, prostředky sledování a vyhodnocování práce systému. Funkce CASE nástrojů této kategorie se často překrývají s funkcemi obecných vývojových prostředí.

Post CASE – podporuje organizační činnosti (zavedení, údržbu a rozvoj IS).

2.4 Komponenty CASE nástrojů

Současné CASE nástroje již zpravidla nepředstavují pouze jednoúčelové softwarové prostředky, plnící otrockým způsobem jedinou funkci, nýbrž se dnes již běžně jedná o komplexní integrovanou sadu nástrojů pro podporu vývoje (tzv. Workbench), které podporují přinejmenším 2 fáze životního cyklu vývoje softwarového díla.

Tyto integrované CASE nástroje (dále pouze CASE nástroje) se tedy skládají na základě svých vlastností z dílčích komponent, tedy samostatně použitelných nástrojů a funkčních vlastností, které ve vzájemné kombinaci představují kompletní nástroj pro podporu vývoje. Jaké komponenty jsou v CASE nástroji použity se odvíjí od požadavků, které jsou na něj kladeny (a tedy i jaké požadavky je schopen plnit).

Mezi důležité funkce a vlastnosti CASE patří: [4]

- **Konzistentní grafické ovládací prostředí (GUI)**, tedy jednotný vzhled obrazovek, popisků, tlačítek, jednotné ovládání, použití symbolických ikon apod.
- **Centrální databáze (repository) a správce souborů** pro uchování informací o všech objektech IS (tímto způsobem se zaručí, že informace je použitelná v libovolném dalším kroku projektování).
- **Prostředky verifikace konzistentnosti dat a podpora normalizace dat**
- **Sada podporovaných modelovacích diagramů, jazyků a notací a nástrojů pro jejich transformace**
- **Textový editor pro popis jednotlivých objektů** pro účely technické a uživatelské dokumentace systému, možnost jejího přímého generování ze systému
- **Rozhraní pro připojení rozšiřujících pluginů** za účelem rozšiřování funkčnosti
- **Možnost rychlého návrhu uživatelských obrazovek (UI builder)** včetně simulace vstupů a výstupů
- **Generátor zdrojových programů** (pro případy častého znovupoužití daného kódu)
- **Export / import dat** – pro práci s modely a dokumentací, které byly vytvořeny v jiných programech nebo jsou v jiných programech dále využívány a zpracovávány.

Existují mnohem více způsobů členění (z hlediska interaktivity, fáze projektu vývoje software, atd.). Jednotlivé CASE nástroje mají vlastní řešení jednotlivých komponent, některé mají komponent více, než je uvedeno výše, jiné jich mají méně. Tyto uvedené komponenty vytváří základ pro většinu současně používaných CASE nástrojů a další komponenty dotváří unikátnost konkrétních CASE.

2.5 Metodiky vývoje

Důležitou předností kvalitního nástroje CASE je podpora vývojových metodik. Zpravidla tyto nástroje hlídají dodržování konkrétní metodiky. Cílem těchto metodik je řízení fází životního cyklu vývoje aplikace. Úspěch jejich využití závisí na vlastnostech konkrétní metodiky.

Mezi známé metodiky vývoje software patří:

- RUP
- Select Perspective
- OOSE
- OMT
- MMDIS

Metodika RUP (Rational Unified Process) je metodika zaměřena na tvorbu softwaru na základě průběžných iterací, které zahrnují různé pohledy na to samé řešení a ze kterých si lze vybrat s ohledem na konkrétní projekt. Metodika byla vyvinuta společností Rational, kterou koupila v roce 2003 firma IBM. K samotné metodice, existují i konkrétní produkty (dnes pod střechou IBM).

Select Perspective je sadou nejlepších praktik (best practices) pro vývoj softwaru s podporou procesu návrhu a vývoje komponentových aplikací (Component Based Development). Jedná se o iterativní, inkrementální a adaptivní metodiku, která zahrnuje sadu zdokumentovaných procesních toků (process workflows).

OOSE (Object-Oriented Software Engineering) je objektivě zaměřená metodika vývoje softwaru. Metodika byla vyvinuta v roce 1992 ve společnosti Objectory a „je první metodikou využívající use case k vývoji softwaru“ [6]. Společnost byla později odkoupena firmou Rational a metodika byla zintegrována do RUP.

OMT (Object Modelling Technique) „je jazykem pro objektivě orientované modelování a návrh (design)“ [14]. Jazyk byl vyvinut v roce 1991 a zahrnoval tři typy modelů: „a) Model objektů, b) Dynamický model a c) Funkční model“. Záměrem modelování dle OMT spočívá v: „možnosti simulace před samotnou implementací, komunikaci se zákazníkem, vizualizace a snížení komplexnosti. OMT se stal základem pro pozdější UML“ [14].

MMDIS (Multidimensional Management and Development of Information Systems) je metodika vývoje informačních systémů, která byla vyvinuta na FIS, VŠE. Metodika je zaměřena na komplexní pojetí systému z mnoha hledisek (dimenzí) při jednotlivých etapách projektu. Dimenze, které je třeba sledovat, jsou: funkce/procesy (PRO), data/informace (INF), organizační a legislativní aspekty (ORG), pracovní, sociální a etické aspekty (PRA), software (SW), hardware (HW), uživatelské rozhraní (UR), bezpečnost (BE), ekonomická a finanční (EKO), metody (MET), dokumentace (DOK) a management (MNG).

V prvním kroku lze u nástroje CASE poměrně snadno určit, jaký typ metodiky nástroj CASE podporuje. Zpravidla lze tyto metodiky rozdělit do 2 skupin:

- Strukturované

- Objektové

2.6 Používané typy diagramů

Jak již bylo uvedeno v předchozí kapitole, jednou z nejdůležitějších vlastností CASE nástroje je sada podporovaných diagramů pro modelování softwarových děl ve více fázích životního cyklu a schopnost organizovat a spravovat modely pomocí nich vytvořené. Standardním jazykem pro modelování informačních systémů je jazyk UML. Tento jazyk je považován jako standardní a skládá se z vymezené sady diagramů. Kromě tohoto jazyka se používají také specifické diagramy (např. diagramy pro vytváření datových modelů, atd.), o nichž bude také řeč v této kapitole.

2.6.1 Skupina diagramů UML

Pro vizualizaci, specifikaci, navrhování i dokumentaci programových systémů se v CASE nástrojích nejčastěji využívá objektově orientovaný standard – grafický jazyk UML (Unified Modeling Language), vytvořený a spravovaný organizací Object Management Group. UML definuje řadu typů diagramů a způsob jejich zápisu. Typy se člení na diagramy struktury a diagramy chování, které obsahují podmnožinu diagramů interakcí [7]

- **Diagramy struktury** – popisují systém staticky z hlediska objektů, atributů, funkcí a vztahů
 - Diagram tříd (*Class diagram*) – popisuje strukturu tříd, jejich atributů a vztahů
 - Diagram objektů (*Object diagram*) – nabízí úplný či částečný pohled na instance tříd, konkrétní objekty a hodnoty jejich atributů, existující v systému v určitém čase
 - Diagram vnitřní struktury (*Composite Structure Diagram*) – popisuje vnitřní strukturu třídy a možné interakce přes různé části, porty, konektory
 - Diagram komponent (*Component Diagram*) – vyobrazuje rozložení systému na jednotlivé komponenty, jejich propojení do vyšších celků a softwarových systémů
 - Diagram nasazení (*Deployment Diagram*) – slouží k modelování rozmístění prvků systému do uzlů, často hardwarových prostředků
 - Diagram balíčků (*Package Diagram*) – zobrazuje vazby mezi balíčky tvořícími model
 - Diagram profilu (*Profile Diagram*) – umožňuje definovat vlastní stereotypy, značky a omezení, vytvořený profil lze aplikovat na balíček
- **Diagramy chování** – jsou zaměřeny na dynamickou povahu systému z hlediska spolupráce objektů a změnu jejich vnitřních stavů
 - Diagram případu užití (*Use Case Diagram*) – zobrazuje funkcionalitu poskytovanou systémem z hlediska uživatelů, jejich cílů a způsobu použití systému
 - Diagram činností (*Activity Diagram*) – reprezentuje přesun řízení mezi prvky systému v podobě následnosti podnikatelských činností, nebo činností systému
 - Stavový diagram (*State Diagram*) – popisuje přechody vnitřních stavů entit nebo také podnikových procesů
- **Diagramy interakcí** – zdůrazňují toky dat a přesun řídicí funkce mezi prvky systému
 - Sekvenční diagram (*Sequence Diagram*) – ukazuje posloupnost zpráv zasílaných mezi objekty a životní cyklus objektu v závislosti na zprávách
 - Diagram komunikace (*Communication Diagram*) – zobrazuje sekvenci zpráv mezi objekty či částmi systému a popisuje tím jejich interakce – kombinuje informace z diagramu tříd, případů užití a sekvenčního diagramu, popisuje tak zároveň strukturu i dynamické chování systému
 - Přehled interakcí (*Interaction Overview Diagram*) – druh diagramu činností, v němž uzly reprezentují diagramy interakcí

- Diagram časování (*Timing Diagram*) – používá se pro znázornění činností, které jsou vyvolané během času

2.6.2 Diagram podnikových procesů

Slouží k modelování podnikových procesů, tedy návaznosti činností, rozhodování, událostí, stavů a zdrojů, dohromady tvořících proces. Často se používá notace BPMN a Eriksson-Penker pro zobrazení návaznosti procesů. Pro udržení přehlednosti diagramu se využívá hierarchické členění, v němž může být každá činnost popsána podrobněji novým procesem. Z modelů BPMN lze generovat kód jazyka BPEL pro orchestraci služeb. Pro modelování procesů lze použít také diagramy činností a diagramy případů užití jazyka UML [8]

2.6.3 Entity Relationship diagram

Diagram představuje abstraktní, konceptuální reprezentaci dat systému a patří mezi nástroje strukturované analýzy a návrhu. Skládá se z entit, jejich vazeb a atributů. Při modelování se postupuje od konceptuálního modelu k modelu logickému (nezávislý na konkrétním prostředí) a následně fyzickému (pro konkrétní databázový systém). Tento přístup se označuje jako princip tří architektur. Nástroje umožňují automatický přechod mezi úrovněmi modelu a generování SQL kódu pro daný systém řízení báze dat. Existuje několik notací, lze použít také diagram tříd jazyka UML s příslušnými stereotypy. Některé nástroje podporují také modelování multidimenzionálních datových struktur pro aplikace Business Intelligence [9].

2.6.4 Diagram datových toků

Data flow diagram slouží k modelování funkcí systému. Patří k nástrojům strukturované analýzy a návrhu. Diagram se skládá z procesů (funkcí), datových toků, data storů a terminátorů a pro udržení přehlednosti lze model hierarchicky rozkládat – každou funkci modelovat jako nový diagram. Pro podobné účely se používá diagram funkční dekompozice [10]

2.6.5 Diagramy pro modelování XML

Prostředky, které umožňují grafickým způsobem vytvářet XML Schémata, případně popisy typu dokumentů DTD pro popis struktury a obsahu XML dokumentů.

2.6.6 Diagramy pro CABA, řízení projektů a další

Vedle nástrojů pro počítačovou podporu vývoje informačních systémů existují také nástroje pro podporu řízení podniku Computer Aided Business Engineering, projektové řízení, řízení a provoz informatiky. Mezi diagramy objevující se v takových aplikacích patří například:

- Diagram byznys motivace BMM
- Diagram organizace
- Diagram obchodní komunikace
- Diagram logického modelu
- Diagram aplikační architektury
- Servisně orientovaný diagram
- Diagram technologické infrastruktury
- Projektová šablona
- Zachmanův rámec, FEAF, MODAF, DoDAF, TOGAF

2.7 Přehled dostupných SW nástrojů

Tento přehled poskytuje základní orientaci v nabídce nástrojů podporujících softwarové inženýrství (CASE). U každého z nástrojů uvádíme jeho výrobce, podporované úrovně vývoje, implementované metodiky, používané modely a notace a další významné vlastnosti (podporované programovací jazyky, databázové systémy, nástroje, podpora týmové práce, sdílení rozpracovaných fragmentů, sledování konzistence, automatizace procesů, řízení životního cyklu aplikace).

2.7.1 PowerDesigner 12.5

PowerDesigner je první CASE nástroj, který komplexně pokrývá všechny aspekty rozvoje podniku. Obsahuje nástroje pro obchodně orientovanou procesní analýzu, která umožní identifikovat klíčová místa a funkce podniku jako takového a nabízí také plně integrované prostředí pro datovou a objektovou analýzu informačních systémů. Přitom plně podporuje zavedené přístupy a metodiky jako je Unified Modeling Language (UML) nebo tříúrovňový návrh databáze. Nová verze má plugin do Visual Studia 2005, který přidává různé toolbary a nový typ projektu, který zahrnuje všechny typy diagramů.

Nástroj je dostupný v několika variantách:

- DataArchitect – návrh databáze a řízení požadavků
- Developer – OO modelování, řízení požadavků, UML, Java, C#, VB.NET, XML, PowerBuilder
- Studio – kombinace variant DataArchitect a Developer plus modelování obchodních procesů
- Viewer – slouží ke čtení modelů
- Enterprise Edice (DataArchitect, Developer, Studio) – navíc přístup k centrálnímu úložišti
- PowerDesigner 15 – modely pro Enterprise Architekturu

Nástroj	PowerDesigner 12.5 (15)
Výrobce	Sybase
Zaměření na úroveň vývoje	Analýza a návrh obchodních procesů Datová a objektová analýza a návrh IS Výstup zdrojového kódu Tvorba dokumentace
Implementované notace	metodiky, Objektové i strukturované UML 2.0 MDA (P3A) BPMN Není metodika vývoje IS
Používané modely (diagramy)	BPM (diagram podnikových procesů – diagram procesních řetězců, hierarchie procesů – BPMN) UML (plná podpora diagramů Unified Modelling Language) ERD (diagram entit a jejich vztahů – konceptuální, fyzický, multidimenzionální) XML (podpora DTD a XML Schema) PD 15: Diagram organizace, Diagram obchodní komunikace, Diagram logického modelu, Diagram aplikační architektury, Servisně orientovaný diagram, Diagram technologické infrastruktury, Projektová šablona, Zachmanův rámec, FEAF
Podporované technologie	ebXML, BPEL4WS, podpora SOA (webové služby)

Programovací jazyky	Java, C#, C++, PowerBuilder, XML, VB.NET, Hibernate, EJB3, NHibernate, JSF, WinForm... Obousměrný engineering
Databázové systémy	Oracle, IBM DB/2, MS SQL Server, Sybase, ANSI SQL, AS/400, DB2, Informix, Interbase, Access, MySQL, Postgre, Teradata... Obousměrný engineering Podpora datových skladů
Integrace s nástroji	Eclipse, PowerBuilder, Visual Studio
Další vlastnosti	Generování dokumentace (RTF, HTML, Excel) Generátor kódu z modelu tříd a reverzní generátor modelu Generátor kódu z ERD a reverzní generátor modelu Řízení požadavků Analýza dopadu změn (i mezi modely) Kontrola konzistence, propojení modelů Mapovací editor objektů Centrální úložiště, sdílení a správa verzí modelů, řízení přístupu na úrovni rolí, modelů a submodelů Široké možnosti rozšíření
Cena	\$2 995 – \$7 495 (51 600 Kč – 123 000 Kč) za uživatele

Tabulka 1: Přehled vlastností PowerDesigner 12.5

[7][8][9][10][11]

2.7.2 Oracle Designer 10g

Tento nástroj se dále nevyvíjí. Na místo něj nastoupil JDeveloper pro vývoj aplikací v Javě a návrh webových služeb; BPA Suite pro analýzu, návrh a optimalizaci podnikových procesů, a nástroj (není CASE) BPM Suite pro návrh a rychlé zprovoznění procesů – využívá k tomu SOA Suite. Přičemž změření zůstává stejné – rychlý vývoj aplikací založených na relační databázi.

Projekty jsou rozděleny na modelování systémových požadavků (BPM, ERD, DFD, hierarchie funkcí), generování předběžného designu (generování databáze, obrazovek, výstupů, menu), design a generování kódu a utility (průzkumník objektů, zálohování, správa verzí a závislostí, maticové diagramy, reporty).

Nástroj	Oracle Designer 10g
Výrobce	Oracle
Zaměření na úroveň vývoje	Analýza a návrh obchodních procesů Datová analýza a návrh Výstup zdrojového kódu
Implementované notace	metodiky, Strukturované MDA (P3A) Metodika Custom Development Method
Používané modely (diagramy)	BPM (diagram podnikových procesů – diagram procesních řetězců, hierarchie procesů) ERD (diagram entit a jejich vztahů) DFD (diagram datových toků) Diagram funkční dekompozice CRUD
Podporované technologie	Oracle Forms, Oracle Reports
Programovací jazyky	Oracle Forms, Oracle Reports, PL/SQL, web PL/SQL Obousměrný engineering
Databázové systémy	Oracle, MS SQL Server, DB2, Sybase, ANSI SQL, ODBC

	Obousměrný engineering
	Podpora datových skladů
Integrace s nástroji	Oracle databáze
Další vlastnosti	Generování dokumentace (slabé)
	Generátor kódu z modelu tříd a reverzní generátor modelu
	Generátor kódu z ERD a reverzní generátor modelu
	Návrh uživatelských obrazovek
	Kontrola konzistence
	Maticové diagramy
	Centrální úložiště, zálohování, sdílení a správa verzí objektů, řízení přístupu uživatelů
Cena	\$5000 (86 200 Kč) za uživatele

Tabulka 2: Přehled vlastností Oracle Designer 10g

[7][8][9][10][11][12][13]

2.7.3 Enterprise Architect 7.5

Velmi rozšířená a oblíbená platforma pro modelování, analýzu, návrh, testování a provoz, založená na UML 2.1. Produkt je dostupný v mnoha variantách.

Nástroj	Enterprise Architect 7.5
Výrobce	Sparx Systems
Zaměření na úroveň vývoje	Analýza a návrh obchodních procesů Datová a objektová analýza a návrh Výstup zdrojového kódu Tvorba dokumentace Testování Provoz a údržba Projektové řízení
Implementované notace	metodiky , Objektové i strukturované UML 2.1 BPMN MDA (P3A) Není metodika vývoje IS
Používané modely (diagramy)	BPM (diagram podnikových procesů – diagram procesních řetězců, hierarchie procesů – BPMN) UML (plná podpora diagramů Unified Modelling Language) ERD (diagram entit a jejich vztahů – konceptuální, fyzický, multidimenzionální) XML (podpora XML Schema, WSDL, BPEL) Byznys Rules modely Zachman Framework, MODAF, DoDAF, TOGAF
Podporované technologie	Oracle Forms, Oracle Reports
Programovací jazyky	C, C++, Java, C#, VB.Net, Visual Basic, Delphi, PHP, Python, ActionScript, CORBA, Ada, Verilog, VHDL, System C, VB.Net Obousměrný engineering
Databázové systémy	Oracle 9i a 10g, MS SQL Server, MySQL, Access, PostgreSQL, DB2, InterBase, Informix, Ingres, Sybase ASE, ASA, Firebird Obousměrný engineering
Integrace s nástroji	Visual Studio .NET, Eclipse
Další vlastnosti	Generování dokumentace (RTF, HTML) Generátor kódu z modelu tříd a reverzní generátor modelu Generátor kódu z ERD a reverzní generátor modelu Generování modelů závislých na platformě – DDL, Java, C#, EJB, XSD, JUnit, NUnit, WSDL Podpora XML schémat, obousměrné generování (XSD – UML) Podpora WSDL, obousměrné generování (WSDL – UML) Podpora BPEL, generování z modelů BPMN 1.1 Podpora testování a simulace, provozu a údržby Podpora projektového řízení Řízení požadavků Kontrola konzistence Centrální úložiště (databáze nebo sdílený síťový disk .EAP), sdílení, verzování a porovnávání objektů (Subversion, CVS a SCC), řízení přístupu uživatelů Možnost připojit plug-iny
Cena	\$95 – \$849 (1 600 Kč – 14 600 Kč) za uživatele

Tabulka 3: Přehled vlastností Enterprise Architect 7.5 [8][13]

2.7.4 Select Architect 7.0

Nástroj orientující se na vývoj vícevrstvých aplikací, založený na komponentách. Volitelně jako nadstavbu lze dokoupit tyto nástroje:

- Select Solution for MDA – transformace analytického modelu do modelu návrhu a vzájemná synchronizace těchto modelů
- Select Asset Manager – správa komponent
- Reviewer – kontrola správnosti a kompletnosti
- Synchronizery kódu – synchronizace kódu s modelem pro programovací jazyky C++, C#, Java, Visual Basic, PowerBuilder a Delphi

Nástroj	Select Architect 7.0
Výrobce	LBMS
Zaměření na úroveň vývoje	Analýza a návrh obchodních procesů Datová a objektová analýza a návrh Výstup zdrojového kódu Tvorba dokumentace
Implementované notace	metodiky, Objektové UML BPMN MDA (P3A) Metodika LBMS Development Method (Select Perspective)
Používané modely (diagramy)	BPM (diagram podnikových procesů – diagram procesních řetězců, hierarchie procesů – BPMN) UML (neúplná podpora) ERD (diagram entit a jejich vztahů) XML (podpora XML Schema) BMM (byznys motivace – strategie)
Podporované technologie	–
Programovací jazyky	Java, C++, C#, Visual Basic, Delphi a PowerBuilder Obousměrný engineering
Databázové systémy	Access, DB2, Dbase, FoxPro, Informix, Ingress, Interbase, Oracle, SQL Server a Sybase Obousměrný engineering
Integrace s nástroji	Visual Studio .NET, Eclipse
Další vlastnosti	Generování dokumentace (MS Word) Generátor kódu z modelu tříd a reverzní generátor modelu Generátor kódu z ERD a reverzní generátor modelu Podpora XML schémat, obousměrné generování (XSD – UML) Mapování na používané služby a komponenty Podpora simulace a údržby Řízení požadavků Analýza dopadu změn Kontrola konzistence Centrální úložiště (objektové), sdílení objektů, řízení přístupu uživatelů
Cena	\$6 744 (116 000 Kč) v základu za uživatele

Tabulka 4: Přehled vlastností Select Architect 7.0 [7][9][10][11][14]

2.7.5 Rational Software Architect V7.5

Rational Software Architect je komplexní modelovací a vývojové prostředí, které využívá UML pro návrh architektury aplikací psaných v C++ a J2EE a aplikací založených na webových službách. Nástroj je postaven na softwarovém frameworku Eclipse a je zaměřen na model driven development.

Nástroj	Rational Software Architect V7.5
Výrobce	IBM
Zaměření na úroveň vývoje	Analýza a návrh obchodních procesů Objektová analýza a návrh Výstup zdrojového kódu Tvorba dokumentace
Implementované notace	metodiky , Objektové UML 2.0 BPMN 2 MDD Metodika Rational Unified Process
Používané modely (diagramy)	BPM (diagram podnikových procesů – diagram procesních řetězců, hierarchie procesů – BPMN) UML (podpora diagramů Unified Modelling Language) ERD (diagram entit a jejich vztahů) není zahrnut, používá se nástroj InfoSphere Data Architect XML (podpora XML Schema) DoDAF, UPDM, SoaML
Podporované technologie	Java/J2EE, .NET, WSDL, XSD, BPEL, SCA Eclipse, Rational Team Concert, Rational Asset Manager, Rational Requirements Management, WebSphere Business Modeler, Rational System Architect
Programovací jazyky	C++, Java – obousměrný engineering C#, EJB, WSDL, XSD, CORBA, SQL
Databázové systémy	–
Integrace s nástroji	Visual Studio .NET, Eclipse
Další vlastnosti	Generování dokumentace (HTML) Generátor kódu z modelu tříd a reverzní generátor modelu Podpora XML schémat Podpora SOA Podpora Javy Centrální úložiště (objektové), sdílení objektů, verzování (Rational ClearCase, Rational Team Concert, CVS), řízení přístupu uživatelů
Cena	\$4 474 – \$14 539 (76 600 Kč – 250 000 Kč) za uživatele

Tabulka 5: Přehled vlastností Rational Software Architect V7.5 [7][8][9][10][11][15][16]

2.7.6 Další nástroje

Následující tři nástroje, které nemůžeme zařadit mezi klasické CASE nástroje z prostého důvodu: nejsou CASE nástroji a ani si nekladou hlavní cíl se jimi stát. Vybrali jsme tři podle nás i podle výsledku ankety nejobvyklejší vývojová prostředí (IDE), která v základní verzi nebo prostřednictvím pluginu mohou suplovat některou ze základních funkcionalit CASE nástrojů.

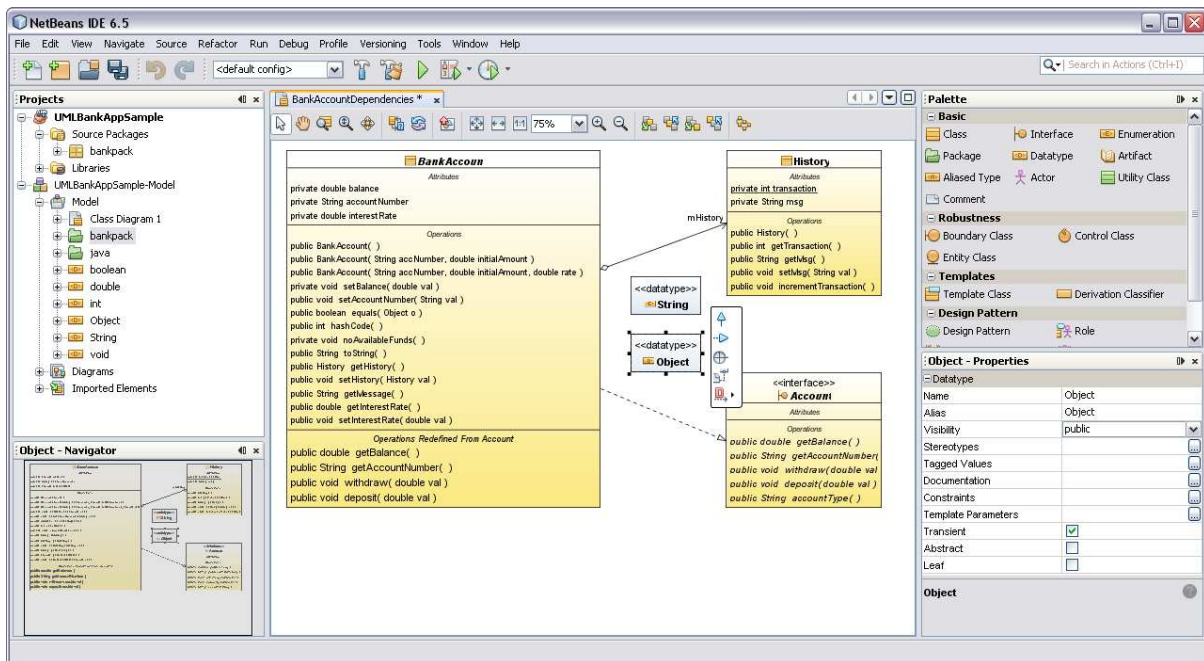
Všechny tři nástroje podporují následující funkcionalitu:

- podpora základních diagramů UML 2.0
- forward i reverse engineering
- centrální úložiště (repozitář)
- generování dokumentace

2.7.7 NetBeans IDE 6.8

Komplexní vývojové prostředí s řadou rozšíření. Jedním z rozšíření je i zásuvný modul UML², který zastává funkcionalitu vhodnou k modelování. Pro jazyk Java možnost forward i reverse engineering. Generování kódu je možné přizpůsobit za pomoci šablon (templates).

Další možností je NetBeans³ rozšířit o placený modul Visual Paradigm SDE⁴, který rozšiřuje funkcionalitu o modelování více než bezplatný modul UML. Zejména generování kódu i pro další jazyky: C++ a PHP.



Obrázek 2: Netbeans IDE 6.5⁵

² <http://netbeans.org/features/uml/>

³ <http://netbeans.org/community/releases/68/>

⁴ <http://www.visual-paradigm.com/product/sde/>

⁵ <http://www.linglom.com/images/Windows/Programming/NetBeans-MSAccess2007/Part-3/1.png>

Nástroj	NetBeans IDE 6.8
Výrobce	Sun Microsystems (Oracle)
Zaměření na úroveň vývoje	Analýza a návrh obchodních procesů Objektová analýza a návrh Výstup zdrojového kódu Tvorba dokumentace
Implementované metodiky, notace	Objektové UML 2.0
Používané modely (diagramy)	BPM (diagram podnikových procesů – diagram procesních řetězců, hierarchie procesů – BPMN) UML (podpora diagramů Unified Modelling Language) ERD XML (podpora XML Schema - XSD)
Podporované technologie	Java/J2EE, .NET, XSD, SOA
Programovací jazyky	Java, C, C++, Python, Ruby, Groovy, PHP, JavaScript, SQL, CSS
Databázové systémy	MSQL, MSSQL, Oracle, InnoDB, IBM DB2
Integrace s nástroji	Visual Paradigm SDE, Kenai
Další vlastnosti	Generování dokumentace (JavaDoc, HTML) Generátor kódu z modelu tříd a reverzní generátor modelu Podpora XML schémat Podpora SOA Podpora Javy Centrální úložiště (objektové), sdílení objektů, verzování CVS, Subversion), řízení přístupu uživatelů
Cena	zdarma

Tabulka 6: Přehled vlastností NetBeans IDE 6.8

2.7.8 Eclipse 3.5.1 (Galileo)

Je jedním z nejpoužívanějších vývojových IDE pro jazyk Java. Eclipse⁶ nabízí vlastní framework k modelování: EMF (Eclipse Modeling Framework)⁷, který společně s GMF (Graphical Modeling Framework) a GEF (Graphical Editing Framework) tvoří základ pro modelování v Eclipse.

Nástroj	Eclipse 3.5.1 (Galileo)
Výrobce	Eclipse Foundation
Zaměření na úroveň vývoje	Analýza a návrh obchodních procesů Objektová analýza a návrh Výstup zdrojového kódu Tvorba dokumentace
Implementované notace	metodiky, Objektové UML 2.0 MDT BPMN2 IMM (Information Management MetaModel) MST OCL (OMG standard)
Používané modely (diagramy)	EMF (Eclipse Modeling Framework) BPM (diagram podnikových procesů – diagram procesních řetězců, hierarchie procesů – BPMN) UML (podpora diagramů Unified Modelling Language) ERD XML (podpora XML Schema)
Podporované technologie	Java/J2EE, .NET, XSD, SOA
Programovací jazyky	Java, C, C++, Python, Ruby, Groovy, PHP, JavaScript, SQL, CSS
Databázové systémy	MySQL, MSSQL, Oracle, InnoDB, IBM DB2
Integrace s nástroji	Topcased's Ecore Editor, Visual Paradigm SDE
Další vlastnosti	Generování dokumentace (JavaDoc, HTML) Generátor kódu z modelu tříd a reverzní generátor modelu Podpora XML schémat Podpora SOA Podpora Javy Centrální úložiště (objektové), sdílení objektů, verzování CVS, Subversion), řízení přístupu uživatelů
Cena	zdarma

Tabulka 7: Přehled vlastností Eclipse 3.5.1 (Galileo)

⁶ <http://www.eclipse.org/>

⁷ <http://www.eclipse.org/modeling/emf/>

2.7.9 Visual Studio 2010 Beta 2 Ultimate

Nejvyšší edice Visual Studia 2010 - Ultimate⁸ obsahuje nástroje pro datové modelování, modelování aplikací i systémů v UML. K aktuálnímu datu se nachází ve verzi beta 2.

Poslední stabilní verzi využitelnou k modelování je Visual Studio 2008 Team System Architecture Edition⁹.

Nástroj	Visual Studio 2010 Beta 2 Ultimate
Výrobce	Microsoft
Zaměření na úroveň vývoje	Analýza a návrh obchodních procesů Objektová analýza a návrh Výstup zdrojového kódu Tvorba dokumentace
Implementované notace	Objektové UML 2.0
Používané modely (diagramy)	BPM UML (podpora diagramů Unified Modelling Language) EDM (Entity Data Model) XML (podpora XML Schema)
Podporované technologie	ASP.NET, XSD, SOA
Programovací jazyky	C#, VB.NET, C++, F#, JavaScript, SQL, CSS
Databázové systémy	MySQL, MSSQL, Oracle, InnoDB, IBM DB2
Integrace s nástroji	Visual Paradigm SDE
Další vlastnosti	Generování dokumentace (HTML, CHM) Generátor kódu z modelu tříd a reverzní generátor modelu Podpora XML schémat Podpora SOA TFS (Team Foundation Server) - sdílení, verzování, kolaborace
Cena	Visual Studio 2010 Beta 2 Ultimate je dostupné zdarma, poslední stabilní verze - Visual Studio 2008 Team System Architecture Edition stojí \$5469 (101 000,-Kč) za licenci

Tabulka 8: Přehled vlastností Visual Studio 2010 Beta 2 Ultimate

⁸ <http://www.microsoft.com/visualstudio/en-us/products/2010/default.aspx>

⁹ <http://www.microsoft.com/visualstudio/en-us/products/teamssystem/default.aspx>

3 Výběr vhodného nástroje CASE

3.1 Cíle společnosti zabývající se vývojem softwaru

Na českém trhu působí mnoho společností, zabývajících se vývojem informačních systémů a softwaru obecně.

Obecně, cílem vývojářské společnosti je vyvíjet software. K jeho podpoře je třeba nějaký CASE nástroj, který společnosti vybírají na základě požadavků a dílčích cílů mezi které můžeme řadit obecně:

- Analýza klientských požadavků
- Optimalizace podnikových procesů
- Návrh informačních systémů a jiných SW produktů
- Implementace SW produktů
- Plánování a realizace změn
- Údržba produktů a aktualizace
- Technická podpora
- Poskytování externích služeb

Z obecného a manažerského hlediska lze mezi tyto cíle zařadit rovněž zrychlení spolupráce na základě společného prostředí, podpora týmové spolupráce a propojení klíčových pracovníků, možnost integrace jednotlivých nástrojů mezi sebou a podpora konkrétních metodologií vývoje.

3.2 Technologie

Důležitým faktorem při výběru CASE nástroje hraje používaná technologie pro vývoj software ve společnosti. Co se týče programovacích jazyků, nástroje CASE podporují dnes různé vývojové technologie, mezi kterými vynikají platforma .NET a JAVA. Obě tyto technologie jsou objektově orientované a lze najít nástroje podporující jednu nebo obojí. V jiných případech lze najít podporu jiných programovacích jazyků (např. Delphi).

Další vlastností nástrojů CASE je podpora datového modelování, kde lze také najít podporu různých typů databází dle zvoleného produktu.

Podpora zmíněných technologií spočívá především v nabídce specifických a konkrétních funkcionalit ke každé technologii a možnost převodu modelu do konkrétního kódu a naopak.

3.3 Metriky využitelnosti CASE nástrojů

K určení efektivity a použitelnosti CASE nástrojů je zapotřebí definovat určité metriky, na jejichž základě lze určit přínosy pro vývoj softwarového projektu.

3.3.1 O metrikách

Softwarové metriky jako takové často používají tvůrci software pro ocenění a odhady složitosti vývoje softwarového produktu (čas, potřebné „know-how“ a z toho plynoucí požadavky na lidské zdroje, atd.) a kvality implementovaných funkcí (programů, modulů, systému...).

Přehled kvalitativních dimenzí softwaru které se při jeho analýze zkoumají a na které existují určité metriky:

- Funkcionalita (vhodnost, přesnost, součinnost, bezpečnost)
- Efektivita (časové chování, využití zdrojů)
- Spolehlivost (vyspělost, odolnost vůči chybám, obnovitelnost)
- Udržitelnost (možnost analýzy, změny, stabilita, testovatelnost)
- Použitelnost (jasnost, učení, ovladatelnost a přitažlivost rozhraní)
- Přenositelnost (adaptovatelnost, instalovatelnost, koexistence, vyměnitelnost)

Ať už se jedná o jakoukoliv metriku tak by ideálně měla mít tyto vlastnosti:

- jednoduchá a přesně definovaná, aby bylo jasno jak se má vyhodnocovat
- co nejvíce objektivní
- jednoduše získatelná (spočítaná, zpozorována, atd.) za „rozumnou cenu/námahu“
- správná – měří opravdu to co má
- robustní – relativně necitlivá k nepodstatným změnám v měřeném produktu/procesu

Jelikož zkoumáme CASE nástroje z hlediska uživatele tak se logicky eliminuje celá řada kvantitativních metrik které se často používají u software:

- Velikost:
 - řádky zdrojového kódu
 - analýza funkčních bodů
 - počet tříd, rozhraní
- Složitost:
 - cyklomatické číslo funkcí/metod
 - počet vstupních a výstupních informačních toků u funkcí/metod
- Kvalita
 - počet chyb na x řádků kódu
 - pravděpodobnost výskytu chyby za určenou dobu (Mean Time To Failure)
- a další...

Výše uvedené metriky uživatele používat nemohou, aniž by podnikli reverse engineering konkrétního CASE nástroje. To by ovšem v drtivé většině případu porušilo licenční podmínky. Celkově jsou aplikovatelné pouze pokud se jedná o open source CASE nástroj. U closed source software tím pádem z hlediska koncového uživatele připadají v úvahu hlavně metriky které se zabývají funkcionalitou, efektivitou a použitelností.

Obecně pomocí metrik se v našem případě snažíme o určení přínosů CASE nástroje pro vývoj softwarového projektu aniž by se tento nástroj nasadil na opravdovém projektu.

3.3.2 Funkcionalita

Funkcionální metrika je kvalitativní – buď zkoumaný produkt danou funkcí disponuje nebo nikoliv. Znázornit se dá třeba do tzv. „feature matrix“ kde po řádcích matice jsou funkce nástrojů, ve sloupcích jednotlivé nástroje a prvky tvoří hodnoty „ano“ nebo „ne“ podle toho zda daný nástroj umí nebo neumí danou věc. Tento relativně primitivní model poslouží k hrubému zkoumání dostupných alternativ na trhu a vyřazení nástrojů které nepřipadají v úvahu z důvodů absence klíčové funkcionality.

CASE nástroje jsou komplexním softwarem který v mnoha případech má tak širokou škálu dostupných funkcí, že to může omezit nasazení feature matrix – ztrácí se přehlednost. Proto je vhodnější vycházet z toho co firma potřebuje od CASE nástroje a ne z toho co nabízí jednotlivé nástroje na svých webových stránkách a pak to dávat dohromady. Pokud se firma z nějakých důvodů (např. zkoumá jenom pár nástrojů nebo v tom horším případě neví co chce) přesto půjde při sestavení matice opačným směrem tak je třeba mít na paměti jeden omezující faktor dané metriky – absence priorit u funkcionality. Na něčem softwarové firmě může jednoduše záležet víc a na něčem méně. Zavedením koeficientů priorit (např. od 1 do 5) u jednotlivých funkcionalit převádí tuto metriku do řad kvantitativních a „odfiltruje funkcionální šum“. Např. může se stát že pro nějakou firmu zabývající se hlavně business modelováním je podpora BPMN velice důležitá, tak se jí přiřadí koeficient „5“. Výsledkem je že každý zkoumaný CASE nástroj má svojí „užitečnost“ pro firmu a ta užitečnost je vyjádřena číselně, tudíž se dá porovnávat, je objektivní (pokud se všichni uvnitř firmy dohodnou na hodnotách jednotlivých koeficientů) a dá se použít jako metrika.

Finance hrají vždy nezanedbatelnou roli při rozhodování o vhodnosti pořizování a nasazování toho či onoho nástroje. Proto by se mohlo úplnost přidáním finanční dimenze (pořizovací cena, cena za podporu...atd.) spočítat poměr „cena/výkon“ jednotlivých nástrojů a řídit se i podle tohoto údaje.

3.3.3 Použitelnost

Měření použitelnosti (usability) se provádí pro zhodnocení efektivity využívání jednotlivých nástrojů. Testy se provádějí s pomocí uživatelů softwarové firmy (tj. hlavně softwarové architektky potenciálního zákazníka CASE nástrojů). Uživatelé jsou obvykle buď přímo pozorovány při interakci se softwarem nebo dotazovány poté (příp. kombinace obojího). V obou případech obvykle vykonávají nějaké akce v aplikaci dle předem napsaných scénářů různé složitosti a detailnosti. Tyto scénáře by měli být založeny na typické práci uživatelů s danou aplikací (např. „*nakresli UML diagram s N třídami, M vazbami, třída C bude mít atributy a,b,c typu T, metody x,y,z vrací void..., třída D...atd.*“). Při pozorování by se měla sledovat obrazovka na které pracuje uživatel a i samotný uživatel (pochopitelně se svým souhlasem). Vyplnění dotazníku doplní informace od uživatele ohledně jeho zkušenosti a zážitku při procházení scénářem (čemu nerozuměl, co se chovalo divně, kde se zasekl...atd.).

V tomto testování je důležité aby CASE nástroj byl dostupný aspoň formou časově omezené trial verze. Uživatelské rozhraní hraje klíčovou roli a může výrazně ovlivnit produktivitu práce uživatelů CASE nástroje. Důležité faktory při zkoumání použitelnosti:

- jasnost – software musí umožnit uživateli pochopit zda je software vhodný a jak se má používat pro konkrétní úkoly a podmínky využití
- učení – software musí umožnit uživateli se naučit aplikovat daný software k jeho účelům
- ovladatelnost – software musí umožnit uživateli ho ovládat a řídit

- přitažlivost – rozhraní softwarového produktu je pro uživatele přívětivé a hezké

Faktor	Název metriky	Cíl metriky	Metoda aplikace	Způsob měření a výpočet	Interpretace
Jasnost	Jasný vstup a výstup	Rozumí uživatel co je po něm žádáno jako vstupní data a jaký výstup mu na to poskytuje software?	Testování uživatele. Pozorování a/nebo dotazník. Spočítat kolik vstupních a výstupních datových položek uživatel chápe a porovnat to s celkovým počtem dostupných položek.	$X = \frac{A}{B}$ <p>kde A = počet vstupních a výstupních datových položek kterým uživatel rozumí správně. B = počet vstupních a výstupních datových položek dostupných z uživatelského rozhraní během testování.</p>	$0 \leq X \leq 1$ Čím blíže je X k 1, tím lépe.
Učení	Jednoduchost učení	Jak dlouho trvá uživateli naučit se používat nějakou funkci?	Pozorování uživatele při testování.	T = střední doba než se uživatel naučí používat funkci správně.	$0 < T$ Čím menší je T, tím lepší.
Ovládatelnost	Srozumitelné chyby	Z jaké části chybových stavů dokáže uživatel zvolit správnou cestu k nápravě situace?	Pozorování uživatele při testování.	$X = \frac{A}{B}$ <p>kde A = počet chybových situací při kterých uživatel zvolil správnou cestu k vyřešení problému B = celkový počet chybových situací během testování.</p>	$0 \leq X \leq 1$ Čím blíže je X k 1, tím lépe.

Přitažlivost	Přizpůsobivost vzhledu uživatelského rozhraní	Z jaké části lze přizpůsobit elementy rozhraní ke spokojenosti uživatele?	Pozorování uživatele při testování.	$0 \leq X \leq 1$ Čím blíže je X k 1, tím lépe.
				$X = \frac{A}{B}$ kde A = počet elementu rozhraní které lze upravit ku přání uživatele B = celkový počet elementu který uživatel přeje upravit

Tabulka 9: Metriky použitelnosti

3.4 Problémy začínajících firem v oblasti CASE

U partnerských společností, stejně tak jako u dalších malých a středních společností zabývajících se vývojem softwaru. Bylo zjištěno, že společnosti čelí určitým potížím, které brzdí jejich rozvoj a uplatnění na českém i evropském trhu.

Mezi nejvýraznější překážky a problémy patří:

- Softwarové společnosti, které vyrůstají samy bez pomoci zahraničního kapitálu a know-how, nemají v převážné většině přístup ke znalostem a nástrojům potřebným pro tvorbu projektů většího rozsahu.
- Potřebné znalosti a nástroje z oblasti analýz a způsobů řízení softwarových projektů jsou pro tyto společnosti finančně nesnadno dostupné; tyto firmy pak vykazují nižší efektivitu při vývoji komplikovanějšího software, než jejich konkurenti z řad velkých softwarových firem.
- Pro společnosti uvedeného rozsahu jsou nedostupní mj. také experti, kteří disponují odpovídajícími znalostmi a mohli by nové metody do firem zavést.
- Současná nabídka vzdělávacích aktivit je zaměřena na poskytování krátkodobých vzdělávacích kurzů, které neumožňují nabytí praktických zkušeností, potřebných pro zavádění standardizovaných, celosvětově používaných metod do praxe.
- Skutečnost, že softwarové společnosti nevyužívají standardizované metody vývoje aplikací, u nich znemožňuje zavedení systémů řízení kvality podle ISO 9000, požadované při uplatnění na zahraničních trzích.

3.5 Perspektivy

Jak již bylo zmíněno, malé firmy zabývající se vývojem programového vybavení, si nemohou dovolit koupit drahé nástroje CASE a ani nejsou schopny zaplatit zavedení metodiky vývoje softwaru. Tuto skutečnost řeší několika různými způsoby. Buď nemají žádnou strategii a vše řeší ad-hoc nebo se v lepším případě snaží, zavést bezplatnou metodiku, soubor pravidel nebo určitý framework, který zanese do firmy zabývající se vývojem softwaru určitý řád. Jedním z takových open-source frameworků, který umožňuje modelování a synchronizaci kódu z/do modelu je právě dále zmíněný Eclipse Modeling Framework.

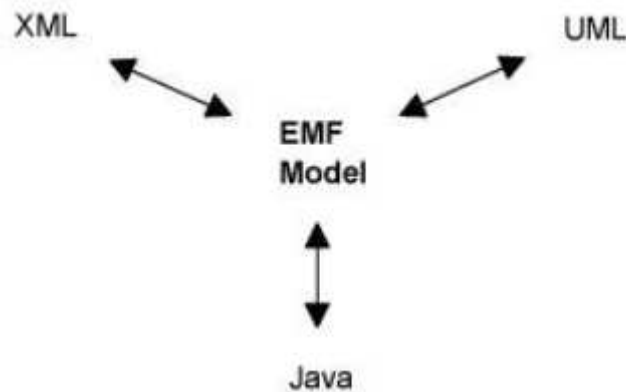
Doufáme, že do budoucna bude přibývat počet firem, které se budou snažit zanést do vývoje softwaru řád a předem definovaná pravidla, a že zároveň bude růst i počet a kvalita dosažitelných metodik malým firmám ve vývoji programového vybavení.

V budoucnu očekáváme následující trendy ve vývoji nástrojů CASE

- Podpora nástrojů CASE ve vývoji software v architektuře Cloud computingu
- Podpora celého životního cyklu aplikace nástrojem CASE
- Podpora většího počtu rozmanitých metodik vývoje
- Možnost přistupovat na dálku k modelům a jejich editaci pomocí běžně dostupných prostředků (Internet + webový prohlížeč)
- Kvalitnější podpora generování kódu z modelů a obráceně (forward a reverse engineering)

1.1.1 Eclipse Modeling Framework (EMF)

EMF je Eclipse modelovací framework a nástroj na generování kódu podporující vývoj v jazyce Java na základě jednoduchých definic modelu. EMF sjednocuje tři důležité technologie: Java, XML a UML. Model může být definován použitím jazyka UML, XML Schema nebo specifikován anotací abstraktního rozhraní jazyka Java. EMF zjednodušuje spolupráci mezi vývojáři a SW inženýry.[19]

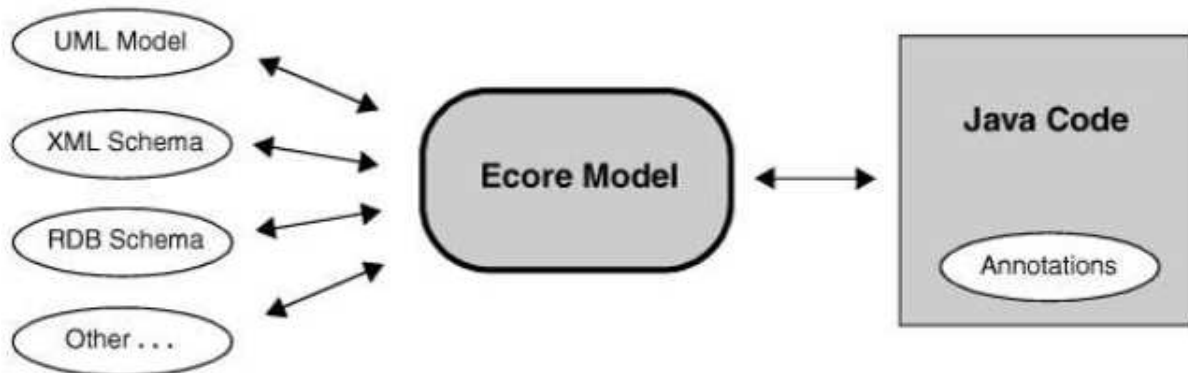


Obrázek 3: Schéma Eclipse Modeling Framework (EMF)[19]

EMF stojí někde na pomezí mezi: „Musím všechno zanalyzovat a namodelovat!“ a „V životě jsem nepotřeboval CASE nástroj a nevím co je UML!“ respektive Model Driven Architecture (MDA) a agilními metodikami.[19]

Ecore – EMF Meta model

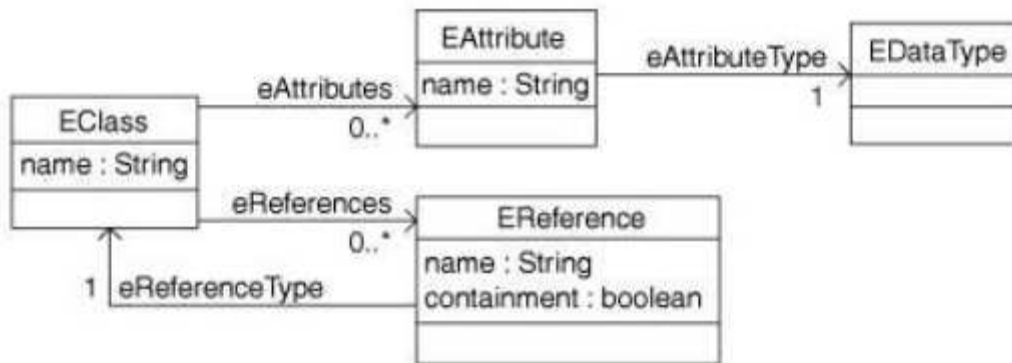
Model, který představuje modely v EMF se nazývá Ecore. Ecore je svým vlastním EMF modelem, je tedy metamodelem. Proč jsou popsány EMF modely právě modelem Ecore a nikoliv jazykem UML? Jazyk UML má daleko širší záběr a je abstraktnější než by bylo potřeba, proto si Eclipse Foundation vytvořilo vlastní framework.[20]



Obrázek 4: Schéma - Ecore EMF Meta model[20]

Nástroje pro vytváření EMF

- **Eclipse Modeling Tools** - <http://www.eclipse.org/downloads/> - freeware Eclipse IDE včetně EMF, GMF, MDT, UML2, atd.
- **Topcased's Ecore Editor** - <http://www.topcased.org/> - freeware Eclipse Plugin
- **Omondo's EclipseUML** - <http://www.omondo.com/> - freeware plugin i samostatná aplikace
- **Soyatec's eUML** - <http://www.soyatec.com/> - freeware Eclipse Plugin, jako samostatná aplikace placené



Obrázek 5: Jednoduchý příklad EMF Ecore Meta modelu[20]

4 Průzkum trhu o využívání CASE nástrojů

Jedním z klíčových cílů této práce je získání aktuálního přehledu o stavu používání CASE v českých firmách. Za tímto účelem byla vytvořena anketa o 20 bodech, která byla rozeslána vybraným vývojářským firmám, z jejíž výsledků jsou vyhodnoceny strukturované závěry.

Zmíněný anketní formulář je umístěn online na:

<http://spreadsheets.google.com/viewform?formkey=dDJVcjNodWZKbGl6QndlN05hTjgzZFE6MA>

Firmy, které se podílely na anketním průzkumu:

- Unicorn Systems, a.s.
- Inexes, s.r.o.
- J.K.R., spol. s.r.o.
- Bellman Group, s.r.o.

Neoficiální cestou byly získány informace i od několika dalších firem, a k jejich odpovědím je rovněž přihlíženo v analýze.

4.1 Cíle průzkumu

Hlavním cílem uspořádané ankety je zmapování současných trendů českých firem v oblasti používání CASE. Každá firma má specifickou interní politiku a má rozdílné zvyklosti v přístupu k CASE. Na základě profilu firem však lze vyhodnotit kategorie českých firem, mající podobné zvyklosti a na tomto základě vytvořit právě 3 kategorie firem: malá, střední a velká firma, zabývající se vývojem software. Klíčovým parametrem pro určení kategorie je počet zaměstnanců a především počet aktivních SW architektů.

Malé firmy nemají žádného či jednoho softwarového architekta, střední firmy mají 2-8 SW architektů a velké firmy mají více než 8 SW architektů. Kromě SW architektů lze na adekvátní pozici považovat také procesního architekta / analytika. V analýze jsou obsaženy informace ze všech 3 zmíněných kategorií.

4.2 Analýza výsledků

Výsledky ankety je vhodné kvůli lepšímu kontextu strukturovat do 5 klíčových bodů na základě relevantnosti. Z těchto bodů lze následně vyvodit mnoho závěrů k celkovému zhodnocení stavu používání CASE na českém trhu.

4.2.1 Trend používání CASE

Hlavní otázkou, kterou si jistě každý položí, je „Používá se vůbec v českých firmách CASE?“. Z ankety jasně vyplývá, že v tomto ohledu silně záleží na velikosti vývojářské firmy. Zažitý vývoj pomocí CASE je zaveden především ve větších firmách.

Jakmile je ve firmě zavedena funkce SW architekta či procesního architekta, předpokládá se, že firma vyvíjí software za pomoci CASE. Ve většině se SW architekti také sami podílí i na implementaci softwaru, tudíž je ocenitelné, pokud CASE nástroj umožňuje také integraci s vývojovým IDE.

Velké firmy mají bohaté zkušenosti s používáním CASE a většina projektů je řízena právě pomocí CASE, i když u některých (malých) projektů nástroje CASE nepoužívají. U středních firem je situace podobná, což neplatí u firem menších, jelikož dbají především na zkušenosti programátorů a intuici. Malé firmy se snaží ušetřit prostředky, jelikož nemají tak vysoký obrát a běžně nevyvíjí příliš komplexní software. Z toho důvodu taková firma ušetří na platech poměrně drahého SW architekta, CASE nástroje a jiných prostředků. Ovšem za cenu ztráty výhod, které CASE nabízí.

4.2.2 Používané nástroje a technologie

Obecně nejpopulárnějším používaným CASE nástrojem ve většině analyzovaných firem se stal Enterprise Architect. Hlavními důvody firem pro nasazení tohoto nástroje se stala cena a funkcionality, plně dostačující na většinu projektů. Ačkoli existují sofistikovanější nástroje jako například PowerDesigner nebo IBM Rational Architect, tyto nástroje jsou příliš drahé a náročné (výpočetně i znalostně). Na modelování business procesů se používá běžně MS Visio, především u malých a středních firem. U velkých firem se na modelování business procesů používají komplexní modelovací nástroje, například Aris Toolset či Erwin. Zákazník si pro interní potřeby může také vyžádat konkrétní nástroj, v němž si přeje realizovat návrh aplikace, což se týká spíše velkých projektů vyvíjených velkými firmami.

Kromě hlavního (standardního) CASE nástroje, používaného na návrh většiny firemních projektů, se často používají i další CASE nástroje. Nasazení alternativních CASE je dáno především nedostatkem funkcionality či omezeními standardně používaného CASE. V takovém případě firmy nepřecházejí na jiný CASE nástroj přímo, jelikož by to bylo časově náročné a pracné transformovat stávající modely do nového CASE nástroje. V tomto případě se i nadále používá standardní CASE nástroj a dodatečné modely se vytváří v alternativním CASE nástroji, nebo se vybere, zda je vhodné dělat v alternativním CASE nástroji celý projekt.

U malých a středních firem, jak již bylo zmíněno, je často z důvodu úspory financí a především kvůli faktu, že SW architekt zastává i funkci programátora, používán CASE nástroj integrovaný ve vývojovém IDE (především Netbeans IDE, Visual Studio).

Většina dotazovaných firem používá programovací platformy .Net a Java, případně programovací jazyk C++, které všechny mají v CASE nástrojích velkou podporu.

4.2.3 Přínosy a výhody

Hlavním přínosem CASE nástrojů je fakt, že se jedná o efektivní týmový komunikační prostředek. Jeho využití zpravidla roste s velikostí týmu, především jako prostředek, s jehož pomocí lze názorným způsobem promítnout změny a strukturu aplikace dalším zúčastněným pracovníkům, čímž zajišťuje zároveň systematičnost práce. Lze jej využít také při řízení rizik. Pokud například klíčový pracovník (programátor) opustí firmu během projektu a na jeho pozici je obsazen nový pracovník (či je přidán další klíčový člen), je oboustranně mnohem snazší mu vysvětlit strukturu aplikace a navázat na proces vývoje pochopením grafického návrhu, než usilovnou studií zdrojového kódu. Zároveň (společně s vhodnými metodikami) lze předejít mnoha chybám v implementaci.

Z hlediska návrhu se jedná zároveň o vynikající prostředek k dokumentaci projektu, který umí generovat výstupy. CASE nástroje mají zpravidla rovněž funkci generování výsledného kódu, která nebývá v praxi využita. Výjimku tvoří generování datové vrstvy, což bývá v praxi velmi často používáno.

4.2.4 Nevýhody a omezení

Pokud zvážíme veškeré přínosy CASE, je zapotřebí přihlídnout rovněž k jejich nevýhodám. Hlavní nevýhody CASE se vztahují především ke komplikovanosti. Současné CASE nástroje jsou někdy příliš robustní a obsahuje mnoho funkcí, které jsou zbytečné a snižují přehlednost daného nástroje, případně zvyšují výkonnostní nároky. Rovněž se to týká příliš složitých mechanismů při nasazování CASE nástrojů, kdy je nutná kompletní změna metodiky a procesů ve firmě.

Dalšími problémy jsou i často zbytečná detailnost notace (po úroveň návrhu tlačítek), velké množství standardů a některé zbytečné typy diagramů.

Poněkud zbytečnou funkcí je generování přímého kódu. Tuto funkci při vývoji používají firmy zpravidla pouze pro generování datových skladů. Přímý aplikační kód generovaný CASE nástroji není přehledný a je těžko udržovatelný, což není pro firmy zabývající se vývojem využitelné.

Neméně podstatnou překážkou je nepřehledné generování dokumentace (zpravidla pouze abecední seznam diagramů s popisky). To značně omezuje efektivitu CASE nástroje jako pomůcky pro tvorbu dokumentace.

4.2.5 Očekávaná zlepšení

Firmy projevují zájem o větší flexibilitu při rozšiřování možností CASE nástroje, například pomocí rozšiřujících modulů, kterými lze rozšířit základní potřebnou funkcionalitu a také o možnost modelování formy dokumentace (šablonování výstupů dokumentace) a efektivní strukturování výstupu, tedy eliminaci nepřehlednosti generování dokumentace zmíněnou dříve.

Firmy by jistě ocenily i jiná zlepšení, nicméně nejsou aktuálně schopny přesněji specifikovat hlubší požadavky na změnu či zavedení nového nápadu. Tato sekce se s rozvojem technologií drastickým tempem mění a není možné vyloučit, že i očekávaná zlepšení již v rámci některých jiných produktů nebyla provedena.

4.3 Zhodnocení

V rámci ankety se podařilo splnit vytyčené cíle a jejím výstupem se stal poměrně přínosný materiál, sloužící jako zdroj informací o pojetí CASE v českých podmínkách a jeho významu. Nejdůležitějším poznatkem této analýzy je bezesporu přístup malých firem k používání CASE nástrojů. Navzdory komplikacím a nákladům by se malé firmy měly zamyslet nad přínosy CASE a zvážit okolnosti, vztahující se k tomuto fenoménu, za účelem zavedení softwarového inženýrství s plnou podporou CASE.

5 Závěr

V rámci této práce se podařilo naplnit všechny zamýšlené cíle a vytvořit uspokojivý dokument, popisující současný stav používání CASE nástrojů v českých vývojářských firmách. Zároveň tento dokument tvoří jakýsi úvod do terminologie stále dramatičtější se rozšiřujícího odvětví, jakým je CASE, potažmo softwarové inženýrství.

Cílem této práce nebylo vyzdvihnout a doporučit ideální CASE nástroj pro vývoj software, nýbrž zmapovat vlastnosti současných nástrojů v kontextu požadavků firem. Za tímto účelem byla vyhotovena anketa, která přinesla poznatky na poli domácího trhu práce v oblasti vývoje informačních systémů. Výsledky této studie otevírají možnosti pro další debaty a úvahy, kterak zlepšit současný stav a informovat nově příchozí účastníky o moderních trendech v této oblasti.

Použitá literatura:

- [1] CASE nástroje. *Wikipedia*. [Online] http://cs.wikipedia.org/wiki/CASE_n%C3%A1stroje.
- [2] CASE. *Wikipedia*. [Online] <http://cs.wikipedia.org/wiki/CASE>.
- [3] **Jirovský, Juhás, Krejčík, Pelcl, Zaplatílek** . Použití CASE/CABE pro řízení workflow ve firmě. [Online] 3. Únor 2008. [Citace: 2. 12 2009.] http://www.panrepa.org/CASE/jaro2007/case_v_workflow_jaro2007.pdf.
- [4] **Bc. František Mencl**. Návrh a realizace internetového obchodu v ASP.NET. [Online] 3. Únor 2008. [Citace: 2. 12 2009.] http://is.bivs.cz/th/10727/bivs_m/Diplomka_Mencl.txt.
- [5] **Jaroslav Procházka**. Nástroje CASE? Co? Proč? Jak?. [Online] 27. 05. 2004. [Citace: 2. 12 2009.] <http://www.dbsvet.cz/view.php?cisloclanku=2004052702>.
- [6] **Michal Beneš**. Přehled OO metodik a notací. [Online] 03.02.2008. [Citace: 2. 12 2009.] <http://objekty.vse.cz/Objekty/MetodikyANotace-NastrojePrehled>.
- [7] **Rydval, Slávek**. Kreslítka pro krycím jménem CASE. *NaWEBka*. [Online] 1. Srpen 2005. [Citace: 2. 12 2009.] <http://www.rydval.cz/phprs/view.php?cisloclanku=2005123135>.
- [8] **Vild, Vojtěch**. Principy objektově orientované analýzy. [Online] 2006. [Citace: 2. 12 2009.] http://is.muni.cz/th/39367/fi_m/DP_Vild.txt.
- [9] **Příkryl, Martin**. Srovnání CASE nástrojů. [Online] Leden 2001. [Citace: 4. 12 2009.] <http://www.prikryl.cz/cze/htmlseminarka.php?id=case>.
- [10] **Beneš, Michal**. Přehled některých programů pro modelování. *Objekty - Metodiky a notace*. [Online] 3. Únor 2008. [Citace: 2. 12 2009.] <http://objekty.vse.cz/Objekty/MetodikyANotace-NastrojePrehled#Oracle>.
- [11] —. Srovnání možností vybraných programů pro modelování. *Objekty - Metodiky a notace*. [Online] 3. Únor 2008. [Citace: 3. 12 2009.] <http://objekty.vse.cz/Objekty/MetodikyANotace-NastrojeSrovnani>.
- [12] **Oracle**. Oracle Designer 10g Release 2. *Oracle*. [Online] Oracle. [Citace: 3. 12 2009.] <http://www.oracle.com/technology/products/designer/index.html>.
- [13] **Sparx Systems**. Enterprise Architect. *Sparx Systems*. [Online] Sparx Systems. [Citace: 4. 12 2009.] <http://www.sparxsystems.com/products/ea/index.html>.
- [14] **LBMS**. Select Architect. *LBMS*. [Online] LBMS. [Citace: 4. 12 2009.] <http://www.lbms.cz/Nastroje/Select-Architect/index.html>.
- [15] **IBM**. Rational Software Architect for WebSphere Software. *IBM*. [Online] IBM. [Citace: 5. 12 2009.] <http://www-01.ibm.com/software/awdtools/swarchitect/websphere/>.
- [16] **Wikipedia**. IBM Rational Software Architect. *Wikipedia*. [Online] [Citace: 3. 12 2009.] http://en.wikipedia.org/wiki/IBM_Rational_Software_Architect.
- [17] **Jacob, Bart, et al**. *Introduction to Grid Computing*. s.l. : IBM International Technical Support Organization, 2005. p. 262. 0738494003.

- [18] **Plaszczak, Pawel a Wellner Jr., Richard.** *Grid Computing: The Savvy Manager's Guide.* místo neznámé : Morgan Kaufmann, 2005. str. 288. 0127425039.
- [19] **William Moore,** Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework. [s.l.] : IBM Redbooks, 2004. 250 s. ISBN 978-0-7384-5316-3.
- [20] **Dave Steinberg,** EMF: Eclipse Modeling Framework. 2nd revised and updated edition. [s.l.] : Addison-Wesley Professional, 2008. 744 s. ISBN 978-0-321-33188-5.