Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Katedra informačních technologií

Studijní program: Aplikovaná informatika

Obor: Informační systémy a technologie

Diplomant:	Bc. Petra Hradecká
Vedoucí diplomové práce:	Prof. Ing. Václav Řepa, CSc.
Oponent diplomové práce:	Ing. Oleg Svatoš

Metodika pro výběr a nasazení CASE nástrojů

školní rok 2007/2008

Vysoká škola ekonomická — Katedra informačních technologií

Fakulta informatiky a statistiky — Školní rok 2007/2008

ZADÁNÍ DIPLOMOVÉ PRÁCE

Jméno : **Petra Hradecká**

Obor: : Informační systémy a technologie

Vedoucí katedry Vám ve smyslu nařízení vlády o státních závěrečných zkouškách a státních rigorózních zkouškách určuje tuto diplomovou práci:

Téma : Metodika pro výběr a nasazení CASE nástrojů

Osnova:

1. Úvod
    a. Vymezení oblasti
2. Vývoj CASE nástrojů
3. Současná situace na trhu CASE nástrojů
4. Možné směry vývoje v budoucnosti
5. Metodiky pro tvorbu IS
    a. Přehled současných metodik
    b. Návrh metodiky pro výběr CASE nástroje a její aplikace
6. Závěr
    a. Poznatky a přínosy

Seznam literatury :

1. Řepa, V. Podnikové procesy: procesní řízení a modelování. Praha, Grada Publishing, 2008. ISBN 80-247-1281-4
2. Řepa, V. Analýza a návrh informačních systémů, Ekopress, Praha, 1999. ISBN 80-86119-13-0
3. Řepa, V. Chlapek D. Materiály ke strukturované analýze. Praha, Vysoká škola ekonomická, 1997. ISBN 80-7079-260-4
4. Buchalcevová, A. Metodiky vývoje a údržby informačních systémů. Praha, Grada, 2005. ISBN 80-247-1075-7
5. Dohnal, J. Pour, J. Architektury informačních systémů, Ekopress, Praha, 1997. ISBN 80-86119-02-5
6. Chlapek, D., Řepa, V., Stanovská, I. Techniky a nástroje vývoje IS, Praha, 1999. Skripta VŠE Praha, ISBN 80-245-0005-1
7. Voříšek, J. Strategické řízení informačního systému a systémová integrace. Praha, Management Press, 1999. ISBN 80-85943-40-9
8. Vodáček, L. Rosický, A. Informační management, pojetí, poslání a aplikace. Praha, Management Press, 1997. ISBN 80-85946-35-2
9. Krbilová, I. Nagy, P. Peniak, P. Informačné systémy, Žilina, 2003. Skriptum, EDIS, http://fel.utc.sk/~nagy/IS/SKRIPTA.html
10. Hlavenka, J. a kolektiv. Výkladový slovník výpočetní techniky a komunikací. Praha, Computer Press, 1997. ISBN 80-722-6023-5

Vedoucí diplomové práce: doc. Ing. Václav Řepa, CSc.

Datum zadání diplomové práce: červenec 2007

....

Vedoucí katedry Děkan

V Praze, dne 30.6.2008

Prohlášení

Prohlašuji, že jsem diplomovou práci zpracovala samostatně a že jsem uvedla všechny použité prameny a literaturu, ze kterých jsem čerpala.

V Praze dne 25. června 2008

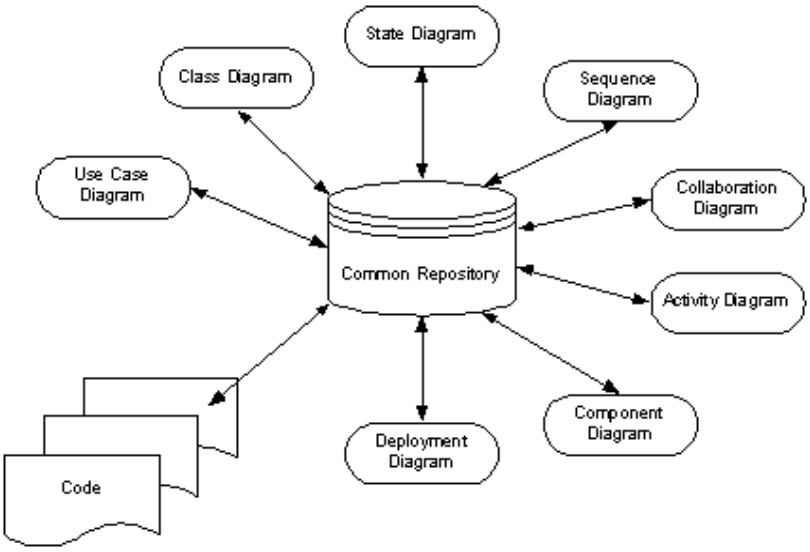
.....

podpis

Poděkování:

Ráda bych tímto poděkovala panu Prof. Ing. Řepovi, CSc., který mi umožnil zpracovat toto téma a za ochotu, s jakou mi vycházel vstříc.

Dále bych ráda poděkovala Ing. Tomáši Šalamonovi, M.B.A. za cenné připomínky a Bc. Petru Tománkovi za jeho podporu při psaní této práce.



http://www.ascilite.org.au/conferences/perth04/procs/armarego.html

Metodika pro výběr a nasazení CASE nástrojů

Petra Hradecká

1. Abstrakt

Cílem této diplomové práce je vytvoření vlastní metodiky pro výběr CASE nástroje.

Práce se skládá ze dvou částí. První část se věnuje softwarovému inženýrství, metodikám pro vývoj IS a jejich kategorizaci, CASE nástrojům a metamodelování. Dále se zabývá metodickým rámcem IS/ICT a aktuální situací na trhu CASE nástrojů.

Druhou část tvoří samotná metodika pro výběr CASE nástroje. Jedná se o otevřenou metodiku, která nabízí doporučení postupů pro výběr CASE nástroje, které jsou odvozeny z praxe, z best practices získaných z dostupných zdrojů a z jiných metodik zabývajících se podobným tématem.

Hlavní přínos této práce spočívá v navržení metodiky pro výběr CASE nástrojů. Proces výběru nástroje jsem rozčlenila do jednotlivých bloků, které následně podrobněji rozebírám. Metodika řeší obě možné výchozí situace, kdy daná organizace již stávající CASE nástroj vlastní a situaci, kdy dosud žádný takový nástroj nepoužívá.

Metodika doporučuje intenzivní spolupráci se zaměstnanci v průběhu procesu výběru a jejich zapojení do rozhodování. To je podstatné z hlediska úspěchu následné implementace a akceptace CASE pracovníky firmy.

Metodika zdůrazňuje, že je nutné si uvědomit odlišnost organizačních struktur, charakteru a rozsahu projektů, rozdílné nároky a požadavky na řízení projektů a také různou úroveň zralosti jednotlivých procesů.

2. Abstract

The goal of this master's thesis is to develop a methodology for CASE tool selection.

This work consists of two parts. The first part is about software engineering, methodologies for IS development and its categorization, CASE tools and meta-modeling. The rest of the part is about methodological framework and current situation on the CASE tools market.

The second part contains the methodology for selection of CASE tool. It is an open methodology and it offers a set of recommendations and procedures that are derived from best practices gathered from available resources and other similar methodologies.

The main idea of this work is to devise a methodology for the CASE tools selection. The selection process is divided into several different segments that are analyzed in detail. The methodology distinguishes between both situations when the company either owns an existing CASE technology or it does not use CASE technology yet.

The methodology suggests intensive internal communication during the selection and emphasizes the involvement of future users in the process. It is important to increase chance for smooth implementation and successful acceptance of the CASE in the company.

The importance of considering differences in structure, character and the scope of projects, requirements for project management and maturity level of processes during the CASE tool selection process is highlighted.

3. Úvod

Žijeme v turbulentní době. Dříve nepředstavitelná rychlost, s jakou se věci v současně době vyvíjejí, s jakou se mění preference spotřebitelů a jejich požadavky a s jakou nové produkty nahrazují ty zastaralé, je dnes realitou. Tato rychlost, s jakou probíhají také všechny výrobní a obchodní procesy, by nebyla možná bez podpory počítačů.

Cílem této práce je navržení vlastní metodiky pro výběr a nasazení CASE nástrojů, protože neexistuje dosud žádná metodika, která by tento problém aktuálně řešila. Stávající metodiky jsou již zastaralé a nevyhovují, neboť byly vytvořeny v určité fázi technologického vývoje, která je dnes již překonaná.

Při psaní práce budu vycházet převážně ze zahraniční literatury a informačních zdrojů. Dále prakticky prozkoumám dostupné CASE nástroje, které jsou na trhu, s cílem jejich vyhodnocení.

Práce bude členěna následovně: v první, teoretické části se budu zabývat softwarovým inženýrstvím, jehož předmětem zájmu je vývoj, nasazení a údržba softwaru. Vzhledem k tomu, že se má práce opírá o obsáhlou terminologii, dále se budu věnovat definici této terminologie a představím základní myšlenky životního cyklu projektu, možné standardy pro lepší uchopení problematiky CASE nástrojů. Poté budu zkoumat jednotlivé metodiky dle specifických kategorizací a ráda bych podala důkazy o jejich potřebnosti, výhodách a přínosu. Ráda bych tuto problematiku také obohatila o příklady z praxe. Dále se budu zabývat samotnými CASE nástroji, jejich členěním, historií, současností, přínosy a nedostatky.

Dále chci provést průzkum aktuální situace na trhu CASE nástrojů. Porovnám mezi sebou typické představitele této skupiny nástrojů a sestavím doporučení pro malé, střední a velké firmy.

Hlavním plánovaným přínosem práce by měla být vlastní metodika pro výběr CASE nástroje, která by měla tvořit druhou část práce.

4. Obsah

"Systém není dokonalý, když k němu nelze nic přidat, ale tehdy, když z něho nelze nic odstranit."

Fred Brook

5. Softwarové inženýrství

5.1. Úvod

V této kapitole se blíže zaměříme na to, co vedlo ke vzniku softwarového inženýrství, co softwarové inženýrství přineslo nového pro vývoj dnešních IS a jak souvisí s metodikami, životním cyklem a CASE nástroji, kterými se budeme postupně zabývat v následujících kapitolách.

5.2. Pojem

Softwarové inženýrství je tradiční inženýrskou disciplínou, která se zabývá přístupem k vývoji, nasazení a údržbě softwaru.

5.3. Historie a současnost softwarového inženýrství

Poprvé se tento termín objevil[1] roku 1969 na konferenci NATO a začal být používán v pojmech jako "*návrh shora dolů*", "*modularita*" apod. V 70. letech vzniká jako samostatný obor a objevují se první techniky softwarového inženýrství, jakými jsou například specifikace, návrh, architektura, testování, zajištění kvality, modely životního cyklu apod. V 80. letech spolu se zvyšováním požadavků na SW vznikají metodologie, metodiky, pojem životního cyklu projektu a objektově orientované přístupy k analýze a návrhu. Vznikají požadavky na standardizaci, interoperabilitu, vzájemnou spolupráci softwarových produktů, rozvoj produktových řad, které umožňují znovupoužití a neustálé zlepšování komponentových technologií, architektur a modelů. Vznikají také první CASE nástroje a aplikace pro podporu nasazení, pro údržbu a správu verzí. V 90. letech vzniká požadavek na znovupoužitelnost, metriky pro sledování kvality softwaru a objevují se návrhové vzory.

Přesto, že bychom vznik pojmu softwarové inženýrství mohli datovat již do roku 1969 a v roce 1990 bylo definováno standardem IEEE[2], bylo ve Spojených státech uznáno jako obor teprve v roce 1997.

I když první CASE nástroje vznikly již v 80tých letech, stále má mnoho firem problém s dokončováním projektů s danými finančními zdroji v dané kvalitě a v daném čase. S rozsáhlostí a robustností dnešních systémů je důležité vlastnit nikoliv jakýkoliv CASE nástroj, ale takový, který bude firmě plně vyhovovat. Správným výběrem a používáním CASE nástrojů můžeme docílit omezení chyb a vyvarování se výše zmíněným problémům.

---

[1] Kadlec, V. Agilní programování: Metodiky efektivního vývoje softwaru. Computer Press, Brno 2004, 1. vyd. ISBN: 80-251-0342-0. str. 26

[2] IEEE Standard Glossary of Software Engineering Terminology: IEEE std 610.12-1990 [online]. IEEE 2004. Dostupné na: http://standards.ieee.org/reading/ieee/std_public/description/se/610.12-1990_desc.html

5.3.1. Softwarová krize

Softwarová krize je název používaný pro období, které se projevovala neustálým prodražováním projektů, nízkou kvalitou výsledných produktů, problematickou údržbou a nízkou produktivitou práce při vývoje, což vedlo ke vzniku oboru, který by se těmito problémy zabýval a našel možná řešení v podobě propracovaných inženýrských postupů zahrnujících best practices, které minimalizují rizika a problémy.

Softwarová krize se datuje do druhé poloviny 60. let[3] a Hovězák[4] uvádí, že se *"hardwarové možnosti počítačů prudce zvýšily a programovací techniky však zůstaly na stejné úrovni, jako byly předtím. Hovoříme proto o tzv. softwarové krizi 60. let. Ve stejné době se objevil i pojem strukturované programování. Podle něj by měl na základě dodržování určitých pravidel umět přečíst a upravit počítačový program i kdokoli jiný, nejen jeho původní autor."*

Pokud se znovu ohlédneme do historie softwarového inženýrství, zjistíme, že jeho vznik byl spojen se softwarovou krizí.[5,6] Jejích příčin bylo mnoho a není jednoznačné, co vše ji přesně způsobilo. Důvody jejího vzniku byly především následující[7]:

- Špatná komunikace. Špatná komunikace byla na všech úrovních (zákazník, analytik, vývojář, manažer), tedy problém komunikace v rámci všech zúčastněných na tvorbě IS a komunikace mezi vývojáři a zákazníkem. Vzhledem k tomu, že je komunikace klíčová pro úspěšnost projektu, je základem každé dobré metodiky.
- Nesprávný přístup jednotlivých osob k vývoji. Týká se především vývojářů, kteří se kromě napsání kódu snažili vyřešit problém atypickým způsobem (aby si dokázali, jak jsou dobří). Cílem bylo, aby kód splňoval určitou funkčnost a nebylo důležité, jakým způsobem (třeba již v minulosti použitým) bude daná funkce vyřešena. Další problém spočíval v minulosti v tom, že vývojář měl nejen roli vyhotovitele IS ale také schvalovatele IS, z čehož vyplývá, že to, co považoval vývojář za dokonalý bezchybný IS, zákazníkovi nevyhovovalo a nesplňovalo jeho požadavky a často byl nespokojený zákazník označen za osobu, která mu nerozumí. V dnešní době se využívá systémů pro řízení vztahů se zákazníky (např. CRM) a vývojáři i celé týmy jsou si vědomi, že pracují pro spokojenost zákazníka a ne pro spokojenost vlastní.
- Nesprávné odhady. Vývojové firmy se zabývaly především psaním kódu, které bylo tím nejdůležitějším v rámci celého vývoje IS. Dnes již víme, že nejdůležitější částí je analýza, naplánování veškerých zdrojů (lidských, znalostních, časových, atd.) přesto bychom neměli zapomínat,

---

[3] Dudka, K. Softwarové inženýrství [online]. Dostupné na: http://dudka.cz/studyIUS

[4] Hovězák, V. Historie počítačů. Dostupné na: http://www.kyberman.wz.cz/files/1_Historie_pocitacu.pdf

[5] Dudka, K. Softwarové inženýrství [online]. Dostupné na: http://dudka.cz/studyIUS

[6] Buchalcevová, A. Pavlíčková, J. Pavlíček, L. Základy softwarového inženýrství Oeconomica, Praha 2007, 1. vyd. ISBN 978-80-245-1270-9

[7] Kadlec, V. Agilní programování: Metodiky efektivního vývoje softwaru. Computer Press, Brno 2004, 1. vyd. ISBN: 80-251-0342-0. str. 27 – 30

že na počátku projektu nemají naši zadavatelé, tudíž ani my, veškeré požadavky a informace a je třeba zajistit dostatečné rezervy na tyto „plánované" změny v průběhu celého vývoje IS.

- Špatné plánování. Vypracovat plán projektu (natož analýzu) bylo v minulosti nepředstavitelné, neboť plán projektu, vytvořený vývojářem, se velmi často neshodoval (a někdy často i v dnešní době neshoduje) s přáním zákazníka a projekt se musel „nějak stihnout". Díky empiricky ověřeným příkladům a postupům, které firmy používají ve formě best practices a metodik a také díky analýze IS se dnes tento problém minimalizuje.
- Nízká produktivita práce. Týkalo se především programátorů, kteří upřednostňovali co největší funkcionalitu bez ohledu na priority, co je třeba napsat nejdříve. V případě jakékoliv změny bylo často nutné a nevyhnutelné přepracovat celý kód. V dnešní době je produktivita práce zajištěna např. specializací, používáním nových technologií, zlepšováním procesů a klade se zásadní důraz na metodiky, zajištění komunikace a vzájemné spolupráce.
- Neznalost základních pravidel. Neznalost pravidel při práci na projektu či nedůvěra v ně, může být příčinou problémů. V dnešní době existují známá měřítka pravdy, díky kterým se těmto problémům vyhneme, např.: „*přidáním dalších lidí do zpožděného projektu způsobí velmi často jen další zpoždění.*"
- Podcenění hrozeb a rizik. Absence předcházení problému se postupem času měnila v neřešitelné chyby. Nebyl problém v tom potenciální hrozby vyhledat, ale přikládat jim včas důležitost. V dnešní době existují metodiky, které nám pomáhají najít také skryté chyby, např. metodika provádění analýzy rizik, která rozebírá všechny potenciální hrozby, které danému projektu hrozí. Tyto metodiky se nazývají „*Risk-driven approaches*" neboli metodiky řízené riziky, kdy riziko je nejvíce zohledňovanou hrozbou a na jehož základě se odvíjí vše statní. Příkladem je spirálový model vývoje IS viz. kapitola 8.5.1.9.
- Nezvládnuté technologie. Tento problém je aktuální i v dnešní době, neboť neustále převažuje falešná představa o tom, že zavedením nové technologie se všechny problémy samy o sobě vyřeší. Problém je, že většina firem se nezabývá tím, zda daná technologie vyhovuje právě jí či v lepším případě, zda má podmínky pro to, aby se v dané firmě nasadila (snadná implementace, důvěra lidí v novou technologii a jejich snaha se nové technologii naučit). Přesně tímto problémem se bude zabývat i má práce, ve které se pokusím o tvorbu vlastní metodiky pro výběr CASE nástroje.

Přestože jsme i v současnosti obeznámeni s některými z příčin softwarové krize, chyby většinou neustále opakujeme, neboť si je nedokážeme uvědomit, lépe je rozebrat a následně odstranit, protože je to buď velmi drahé (náročné na čas, na lidské příp. jiné zdroje) nebo jsme toho názoru, že nejde o tak zásadní problém. S takovýmito předsudky ale nemáme šanci v dlouhém období konkurovat na trhu. Iti nejlepší z nejlepších se dopouští některých z výše uvedených chyb a nemůžeme to považovat za jejich selhání či ostudu. Tou by bylo, pokud by daný tým nedokázal problém rychle vyřešit, neboť pro dnešní vývojové firmy platí místo „změna je život", „projekt je změna" a tudíž je třeba se změnám přizpůsobit v rámci celého projektu.

Vzhledem k tomu, že v dnešní době bývá spíše důležité vyvinout daný systém v co nejkratší době, tedy již se nepropaguje výroba „prvotřídně

kvalitního softwaru v rozumné době“ ale v co nejkratší době software s rozumnou funkcionalitou, přičemž rozumnou zde mám na mysli takovou úroveň, kdy daný software zvládá bez problémů všechny zásadní požadavky aplikace, tedy takovou úroveň, kdy jsou umožněny veškeré zásadní procesy nezbytné k fungování IS. Proto se soudobé softwarové inženýrství zabývá především možnostmi agilních metodik, odlehčených metodologií a metodickými rámci, extrémním programováním a v neposlední řadě také CASE nástroji. Dalším aktuálním požadavkem je možnost přizpůsobit SW aktuálním požadavkům, neboť funkcionalita, která byla definována na počátku vývoje, ve většině případů neshoduje s funkcionalitou aktuální a je nutné ji neustále přizpůsobovat měnícímu se prostředí a měnícím se požadavkům, proto je schopnost rychlých změn jedním z nejdůležitějších a nejcennějších požadavků na SW. Jak již bylo řečeno ve většině případů dnes platí, že „projekt je změna.“

5.4. Závěr

Softwarové inženýrství je inženýrskou disciplínou, která se zabývá přístupem k vývoji, nasazení a údržbě softwaru. Její vznik se datuje do roku 1969, kdy se tento pojem objevil poprvé na konferenci NATO a od té doby se začal používat.

Softwarové inženýrství je tradiční inženýrskou disciplínou, která se zabývá přístupem k vývoji, nasazení a údržbě softwaru. Jednou z oblastí zájmu softwarového inženýrství jsou také CASE nástroje, které se postupem času vyvinuly spolu s rostoucími požadavky na projekty. CASE nástroje obsahují metodiky, které obsahují best practices a ověřené postupy a technologie, které umožňují efektivnější řízení.

Důvodem vzniku či minimálně pohnutkou pro vznik daného oboru byla softwarová krize, jejíž příčin jsme se dosud ne úplně vyvarovali a ne vždy je možno se těmto problémům, které byly důsledkem krize vždy vyhnout. Je ale důležité tyto problémy řešit a uvědomit si, že projekt je změna a požadavky na projekt v dnešní době zní v co nejkratší době software s rozumnou funkcionalitou a schopnost rychlých změn.

6. Důležité pojmy

6.1. Úvod

V této kapitole se seznámíme se základními pojmy, jejichž pochopení je nezbytně nutné pro porozumění dalším částem této práce. Jedná se o termíny metodologie, metodika, metoda, technika a nástroj.

Uvedené terminy jsou v oblasti vývoje IS, velmi často nejednotně používány a zaměňovány nejen laiky ale i profesionály. Jde zejména o případ záměny termínů metodologie, metodika a metoda anebo termín technika s termínem nástroj. Z tohoto důvodu jsem se rozhodla tuto kapitolu věnovat správnému vysvětlení pojmů, abych poskytla každému čtenáři základnu pro snadné pochopení dalšího textu.

6.2. Metodologie, metodika, metoda

Pojmy metodologie, metodika a metoda bývají vzájemně zaměňovány a bez podrobnějšího prozkoumání a pochopení jejich významu libovolně používány. Není výjimkou najít text, který se zmiňuje o metodologii a přitom je z jeho obsahu patrné, že se zabývá metodikou.

6.2.1. Metodologie

Metodologie je charakterizována jako *"ucelený systém filosofických a všeobecně vědeckých teoretických principů či vědeckých výpovědí týkajících se způsobů získání poznatků o světě, nebo způsobů vytváření idealizovaného obrazu světa."* Pstružina zde dále uvádí, že metodologie se zabývá tím, *"co používají vědy v procesu poznání."*[8]

Metodologie je také definována jako „*tzv. metodologická analýza, která je oborem filosofie. Analyzuje principy a procedury, kterými probíhá zkoumání v konkrétní disciplíně*“[9], tedy jde o vědní disciplínu, která nějakým způsobem rozebírá metodiky, definuje je atd.[10]

Metodologie je vědní disciplínou, která se zabývá studiem toho, jak se metodiky vytvářejí a jakým způsobem jsou aplikovány. Metodologie tedy studuje způsoby, kterými se řeší problémy. Termín „metodologie“ se velmi často

[8] Pstružina, K. Atlas filosofie vědy (Fond rozvoje MŠMT F5 1588/2002) [online]. Dostupné na: http://nb.vse.cz/kfil/win/atlas1/atlas3.htm (. 8. 2007)

[9] Miller, G. A. Fellbaum, Ch. Tengi, R. Wakefield, P. Langone, H. Haskell B. R. WordNet: lexical database [online]. Princeton University. Dostupné na: http://wordnet.princeton.edu/ (navštíveno 12.8.2007)

[10] Kadlec, V. Agilní programování: Metodiky efektivního vývoje softwaru. Computer Press, Brno 2004, 1. vyd. ISBN: 80-251-0342-0. str. 31

používá v oblasti vývoje IS místo odborného názvu metodika, což vzniklo díky špatnému překladu pojmu „methodology“ z anglického jazyka[11].

6.2.2. Metodika

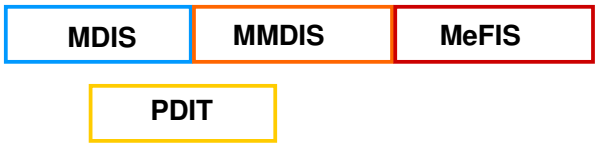
Na metodiku se pohlíží jako na „*systematický přístup k tvorbě informačních systémů, který jasně vymezuje postup a obsah jednotlivých fází vývoje k tvorbě informačních systémů.*“[12]

Pojem metodika vystihuje také následující definice, která uvádí, že se jedná o "*doporučený souhrn principů, konceptů, dokumentů, metod, technik, nástrojů a produktů pro tvůrce informačních systémů, který pokrývá celý životní cyklus informačních systémů (vývoj, údržbu a provoz). Určuje kdo, kdy, co, jak a proč má dělat během vývoje a provozu IS.*"[13] Doporučený princip je ještě rozšířen o pravidla, řízení metod, řízení nástrojů a řízení technik pro tvůrce IS.[14]

Vzhledem ke specifičnosti některých odvětví vyvíjí některé organizace své vlastní metodiky. Jsou jimi například veřejným sektorem vyvíjené a podporované metodiky SSADM (Structured Systems Analysis and Design Method), SDM (System Development Methodology), MERISE, V-MODEL (Vorgehens modell). Řadíme sem také metodiku Evropského společenství Euromethod.

Příkladem firemních metodik může být IE (Information Engineering Methodology), MEIN (Metodologia Informatica), SE (LBMS System Engineering), ORACLE CASE*Method.

Obr. 1: Vývoj metodik na KIT VŠE



Také v České republice jsou vyvíjeny metodiky. Například Katedra informačních technologií na Vysoké škole ekonomické se zabývala od roku 1992[15] vývojem metodiky MDIS a od roku 1995 metodikou PDIT (kromě těchto metodik se např. podílela na vývoji metodiky OOMT[16], který probíhal v letech

---

[11] Wikipedie: Otevřená encyklopedie [online]. Dostupné na: http://cs.wikipedia.org/wiki/Metodologie (navštíveno 12. 8. 2007)

[12] Řepa, V. Analýza a návrh informačních systémů. Ekopress, Praha 1999, ISBN: 80-86119-13-0

[13] Voříšek, J. Strategické řízení informačního systému a systémová integrace. Management Press, Praha 1999, ISBN: 80-85943-40-9

[14] Řepa, V. Chlapek, D. Materiály ke strukturované analýze. Vysoká škola ekonomická, Praha 1997, ISBN 80-7079-260-4. str.18

[15] Buchalcevová, A. Metodiky vývoje a údržby informačních systémů. Grada, Praha 2005, ISBN 80-247-1075-7. str. 40 - 41

[16] Řepa, V. Analýza a návrh informačních systémů. Ekopress, Praha 1999, ISBN: 80-86119-13-0. str. 236

1994 – 1996 a metodiky PDIT[17]). Pokračováním metodiky MDIS je metodika MMDIS (Multidimensional Management and Development of Information System). Autoři metodiky MMDIS v současné době rozvíjejí tuto metodiku ve formě metodického rámce MeFIS (Methodology Framework for IS/ICT Systems).

Obr. 2: Životní cyklus dle metodiky MMDIS[18]

Tento rámec obsahuje vzory metodologií pro vytvoření konkrétní metodiky (jejíž součástí jsou různé typy řešení, typy projektů, definování principů a procesů). MeFIS byl označen jako metodický rámec, protože není sám o sobě metodikou. Jedná se o tzv. „*rodinu metodik postavených na společném základě a specializovaných pro jednotlivé předmětné oblasti, typy projektů podle důležitosti a velikosti, způsoby vývoje apod. Proto byla nazvána metodický rámec.*“ Metodický rámec je chápán „*jako uspořádaná skupina různých, jednotným způsobem popisovaných a klasifikovaných, metodických vzorů. Metodické vzory jsou definovány nejen pro vývoj IS „na zelené louce“, ale i pro rozvoj IS, integraci stávajících systémů anebo nasazení typového aplikačního softwarového vybavení. Metodické vzory jsou specializovány také pro různé problémové domény (např. systémy ERP, Business Intelligence, worflow, elektronického podnikání a další a různé typy projektů.*“[19] Metodickému rámci se budeme později věnovat v rámci kapitoly 11.

[17] Řepa, V. Analýza a návrh informačních systémů. Ekopress, Praha 1999, ISBN: 80-86119-13-0. str. 249

[18] Řepa, V. Analýza a návrh informačních systémů. Ekopress, Praha 1999, ISBN: 80-86119-13-0. str. 18

[19] Buchalcevová, A. Metodiky vývoje a údržby informačních systémů. Grada, Praha 2005, ISBN 80-247-1075-7. str. 71

Metodika je tedy určitý model celého procesu vývoje (kompletního životního cyklu vývoje, o kterém bude pojednáno v kapitole 7). Jedná se o komplexní doporučené postupy a návody pro vývoj IS.[20]

6.2.3. Metoda

„*Metoda je již určitým způsobem, jak provádět určitou věc systematickým způsobem. Je logicky uspořádána, tedy jednotlivé postupy jsou seřazeny v krocích.*“[21]

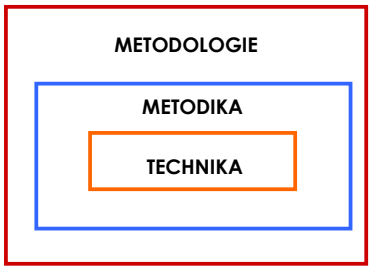
Pojem metoda pochází spojením dvou řeckých slov: meta = po, nad a odos = cesta, neboli vědec by měl ke svým poznatkům docházet vždy po cestě. Lépe než využívat určitou specifickou metodu se užívá spíše obecných metod, neboť najít určitou specifickou metodu je problém.

Metody lze členit na[22]:

- Univerzální. Jsou používány univerzálně všemi obory.
- Obecné. Používány všemi vědami. Pokud je metoda zobecněna, pak postupně dojde k metodě univerzální. V případě, že určitou metodiku není možno zobecnit i přesto může být velmi užitečná a používaná. Obecné metody lze dále rozdělit na pozorování, popis a explanaci, měření a komparaci, experiment, modelování, analýzu a syntézu, indukci a dedukci.
- Specifické. Tato metodika je aplikovaná pro již specifický účel.

Metoda tedy označuje určitý konkrétní postup, který vede k vyřešení určitého dílčího problému. Příkladem mohou být např. metody pro vývoj IS, kterými je např. metoda klasická, Yourdonova, metoda tunel, byrokratická, metoda vodopád, spirála atd., které budou zmíněny v kapitole 8.

Obr. 3: Vztah metodologie, metodika, metoda



Výše uvedený obrázek vyjadřuje vzájemnou souvislost mezi metodologií, metodikou a metodou. Metodologie je věda, která zkoumá, jak jsou vytvářeny

---

[20] Kadlec, V. Agilní programování: Metodiky efektivního vývoje softwaru. Computer Press, Brno 2004, 1. vyd. ISBN: 80-251-0342-0. str. 31

[21] Miller, G. A. Fellbaum, Ch. Tengi, R. Wakefield, P. Langone, H. Haskell B. R. WordNet: lexical database [online]. Princeton University. Dostupné na: http://wordnet.princeton.edu/ (navštíveno 12.8.2007)

[22] Miller, G. A. Fellbaum, Ch. Tengi, R. Wakefield, P. Langone, H. Haskell B. R. WordNet: lexical database [online]. Princeton University. Dostupné na: http://wordnet.princeton.edu/ (navštíveno 12.8.2007)

metodiky, metodika je souhrn metod a návodů tedy obecně platných postupů. Technika je prostředkem, pomocí kterého je uskutečnitelná daná metodika.

6.3. Technika a nástroj

6.3.1. Technika

Technika určuje, jak docílit daného výsledku, jedná se tedy o přesný postup skládající se z konkrétních kroků, které lze využít při řešení určitého problému. Dále se jedná o jednotlivé činnosti a o to, jakým způsobem používat určité nástroje, jaké možnosti můžeme zvážit při rozhodnutích atd. Ve srovnání s metodou je přesnější v závěrech, ale na druhou stranu omezenější co se týče okruhu použití. Technika je tedy jakýmsi návodem, podle kterého se postupuje při provádění specifických činností souvisejících s vývojem IS.[23]

Příkladem techniky je transakční analýza, normalizace datového modelu, Yourdonova funkční analýza, Chennovo datové modelování, Martinovo datové modelování, transformace konceptuálního schéma do logické úrovně návrhu, kanonická procedura, strukturovaná revize atd.[24]

6.3.2. Nástroj

Nástroj *„je prostředkem k uskutečnění určité činnosti v procesu vývoje a provozu IS a prostředkem k vyjádření výsledku této činnosti.“*[25] Nástroj tedy slouží jako softwarová podpora k vytvoření modelů či jiných komponent, kterých je zapotřebí během vývoje IS. Nástroj bývá spojen s konkrétní technikou a vždy formalizuje určité vyjádření. Proto je výhodné, aby byl každý nástroj v co nejvyšší míře automatizován.

Příkladem nástroje je diagram toku dat DFD, diagram entit a jejich vztahů ERD, diagram přechodu stavů STD, diagram řídících toků CFD, matice funkce, matice data, diagram hierarchické struktury, slovník dat atd.

6.4. Vztahy mezi pojmy

Řepa[26] se zmiňuje o rozmanitosti vztahů mezi metodikami, technikami a nástroji i jejich příslušnosti k metodikám. Metodiky se odkazují na jednotlivé metody, technika může patřit k určité metodě nebo může být společnou pro několik metodik. Technika může vyžadovat určitý specifický nástroj nebo se používá jen obecně pro použitelný nástroj a některé z nástrojů mohou být na takové úrovni univerzálnosti, že není důvod, proč je spojovat s určitými technikami to znamená, že se jedná o nástroje, které využívají různé metody.

---

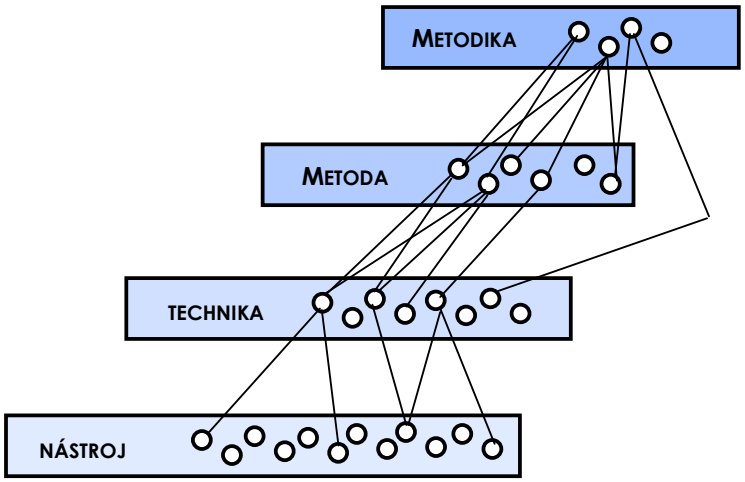
[23] Řepa, V. Chlapek, D. Materiály ke strukturované analýze. Vysoká škola ekonomická, Praha 1997, ISBN 80-7079-260-4. str.20 – 21

[24] Řepa, V. Chlapek, D. Materiály ke strukturované analýze. Vysoká škola ekonomická, Praha 1997, ISBN 80-7079-260-4. str.21

[25] Řepa, V. Chlapek, D. Materiály ke strukturované analýze. Vysoká škola ekonomická, Praha 1997, ISBN 80-7079-260-4. str.21

[26] Řepa, V. Analýza a návrh informačních systémů. Ekopress, Praha 1999, ISBN: 80-86119-13-0. str. 24

Obr. 4: Vztahy mezi pojmy[27]



6.5. Závěr

Metodologie je vědní disciplína, která analyzuje principy a procedury tvorby a aplikace metodik. Metodologie tedy studuje, jakým způsobem lze řešit problémy. Pojem metodologie je velmi často používán v oblasti vývoje informačních systémů místo odborného názvu metodika, což vzniklo díky špatnému překladu pojmu „methodology" z anglického jazyka.

Metodika je určitý systematický přístup, který definuje doporučený souhrn postupů, principů, konceptů, dokumentů, metod, technik a nástrojů pro vývoj IS. Je to tedy určitý komplexní model postupů a návodů pro vývoj kompletního IS.

Metoda je určitý způsob, který dává konkrétní návod na to, jak postupovat při určité dílčí činnosti, tedy co je třeba přesně udělat. Metody se člení na specifické, obecné a univerzální.

Technika určuje způsob, jakým můžeme dojít k danému výsledku. Popisuje tedy přesný postup, který se skládá z jednotlivých kroků, které vedou k požadovanému cíli.

Nástroj je prostředek, pomocí kterého lze uskutečnit určitou činnost v procesu vývoje IS.

Srovnáme-li dané pojmy, metodika slouží jako návod pro zefektivnění procesu v jednotlivých fázích projektu, metodologie zkoumá, jak jsou metodiky vytvářeny a aplikovány a technika je nástrojem, který umožňuje aplikovat danou metodiku.

Vztahy mezi výše uvedenými pojmy mohou nabývat vysoké rozmanitosti, metodiky odkazovat na různé metody, technika může patřit k jedné či více

---

[27] Řepa, V. Metodika vývoje informačního systému s pomocí nástroje Power Designer [online]. Sybase ČR 2006. Dostupné na: http://www.sybase.cz/buxus/docs/Metodika_vyvoje_IS_06_2006.pdf (navštíveno dne 27. 4. 2008). str.7

metodám a vyžadovat jeden či více nástrojů a nástroje mohou využívat různé metody.

7. Životní cyklus projektu

7.1. Úvod

V této kapitole si vysvětlíme pojem životní cyklus, jehož pochopení je důležité pro návaznost dalších kapitol, zabývajících se metodikami. I když je tento pojem v obecné rovině obvykle chápán správně, každý si tento pojem přizpůsobuje podle svých potřeb, což vede ke zvýšení efektivnosti vývoje jednotlivých IS. V této kapitole si také ukážeme několik přístupu k chápání životního cyklu a podíváme se na jeho standardizaci.

7.2. Pojem

Životní cyklus projektu se skládá z jednotlivých etap, které si můžeme představit jako na sebe naskládané kamenné kostky, které takto složené tvoří např. most. Most v našem případě znamená náš cíl projektu, tedy správně fungující IS, a jednotlivé kostky si představíme jako jednotlivé etapy stavby našeho mostu. Nejprve zjistíme, zda se vyplatí most vůbec postavit, tzn. zda přistoupit k jeho realizaci, tedy zjistíme, zda bude opravdu výhodnou investicí, zda jej lidé opravdu potřebují (v případě IS zachycujeme požadavky na systém tzn. jeho funkčnost, design, návaznost na jiné systémy, integraci s ostatními systémy, reakční dobu,…). V případě, že se stavba mostu (tvorba IS) opravdu vyplatí, nastává fáze analýzy, kdy se vytváří nákresy mostu, odhady nákladů a zdrojů (v případě IS jde o konceptuální model, který zachytí danou skutečnost v rámci modelu). V další části vytvoříme přesný nákres mostu (v případě IS vytvoříme implementační implementační model systému, tedy konkrétního návrhu IS), nastává proces stavby mostu (vývoj dané aplikace), tedy samotný vývoj a dále následná implementace a zavedení (nainstalování a spuštění aplikace), kdy je most podroben zátěžovým zkouškám (provedení zátěžových testů na systém) a je spuštěn zkušební přechod (zkušební provoz aplikace). Poté je most uveden do běžného užívání, je v provozu a neustále udržován (udržování IS), jako např. drobné opravy, úklid (změny v IS) a v případě, že hrozí zhroucení, tj. když již ani úpravy a rekonstrukce nemají smysl či se finančně nevyplatí (IS je tak zastaralý a nefunkční, že se nevyplatí opravy a nejlepší možností je vytvořit nový), nastává fáze stažení z užívání (zbourání mostu).

Jak jsme již uvedli na příkladu výše, životní cyklus vývoje IS se skládá z jednotlivých etap (můžeme říci, že i z jednotlivých fází). Jedná se o kompletně pokrytý cyklus vývoje IS od jeho počátků, kdy zvažujeme, zda IS vůbec vyvíjet, přes zachycení jeho požadavků, tvorbu konceptuálního a implementačního (designového) modelu, implementaci, zavedení, testování až po údržbu systému a jeho provoz, v některých případech i stažení systému z užívání.

Každý proces vývoje IS můžeme tedy popsat prostřednictvím životního cyklu. Obecně si zde definujeme jeho základní etapy, kterými jsou:

- zachycení požadavků na systém (týká se funkčnosti, designu, návaznosti na jiné systémy, integrace s ostatními systémy, reakční doby atd.),
- tvorba konceptuálního modelu (zachycení skutečností v rámci modelu),

- tvorba implementačního (designového) modelu (jedná se již o konkrétní návrh IS),
- implementace a zavedení,
- testování,
- udržování systému a provoz,
- stažení systému z užívání.

Pojem životního cyklu je velice rozšířený. I když jej chápe správně jako jednotlivé po sobě jdoucí a na sebe navazující fáze tvorby IS, tedy jako posloupnost kroků, které je třeba vykonat pro dosažení úspěšného vývoje IS, přizpůsobujeme si jej podle svých požadavků a potřeb. To nám umožňuje dosáhnout vyšší efektivity a produktivity při vývoji a mimochodem to životnímu cyklu vůbec neubírá na jeho důležitosti a významu, neboť právě definování životního cyklu a jeho fází bylo základem a nutností pro možnost vzniku metodik.

Metodiky pro řízení a vývoj IS využívají životní cyklus jako svůj základ a definované fáze umožňují podrobněji určit jednotlivé činnosti, na které jsou aplikovány přímo jednotlivé metody. Začátky a konce etap životního cyklu jsou základními klíčovými milníky, podle kterých se řídí vývoj jednotlivých metodik.

Jak již bylo řečeno, ne vždy jsou však fáze životní cyklu striktně definovány. Různí autoři chápou IS z odlišného hlediska, například produktového hlediska, z hlediska vývoje vlastního IS či podle používané metodiky. Uveďme si nyní několik příkladů.

Kendall uvádí tyto fáze vývoje IS[28]:

- identifikace problémů, možností a cílů (stanovení strategie vedoucí k realizaci tvorby IS),
- definování informačních potřeb (proč má produkt vzniknout a jaké informační potřeby uspokojí),
- analýza systémových potřeb (zkoumáme systémové potřeby nezbytné pro tvorbu vlastního IS),
- návrh doporučeného systému (navrhujeme, jak bude systém řešen v dané technologii a jaké postupy budou uplatněny),
- vývoj a dokumentace softwaru,
- testování a zavádění softwaru,
- údržba a hodnocení systému.

Podle Poláka má životní cyklus fáze[29]:

- zadání,
- analýza,

[28] Kendall, K.E. Systém Analysis and Design. Prentice Hall, 1991

[29] Polák, J. Merunka, V. Carda, A. Umění systémového návrhu. Grada, Praha 2003, ISBN: 80-247-0424-02

- návrh,
- implementace,
- testování a provoz.

Podle Řepy[30] vychází životní cyklus z dané metodiky a jako doporučení uvádí, že je dobré pro každou metodiku stanovit:

- cíl etapy (proč etapa má být provedena a co je jejím výsledkem),
- účel a obsah etapy (popis role etapy v celém vývoji systému, shrnutí činností prováděných v etapě),
- předpoklady zahájení etapy (kdy je možné začít s pracemi v rámci dané etapy),
- kritéria ukončení etapy (kdy je možné prohlásit etapu za ukončenou),
- klíčové dokumenty etapy (seznam dokumentů, vyprodukovaných nebo upravených v dané etapě, které musí být schváleny vedením projektu),
- kritické faktory etapy (na co je třeba v etapě dávat největší pozor, faktory, které mohou způsobit problémy při vývoji),
- činnosti etapy (seznam a popis činností, které se v etapě provádějí),
- návaznosti činností v etapě (graficky vyjádřená návaznost a souběžnost provádění jednotlivých činností etapy).

Také se zmiňuje o tom, že každá etapa se skládá z činností a zdůrazňuje, že pro každou činnost by měla existovat metodika, kterou je činnost popsána. Metodika by měla obsahovat:

- cíl činnosti,
- postup (jednotlivé kroky činnosti). Kroky určují, co je třeba udělat v rámci příslušné činnosti, mohou probíhat i opakovaně,
- vstupy (podklady). Uvést, z jakých dokumentů a dalších materiálů se v dané činnosti čerpá, na jaké dokumenty se navazuje,
- výstupy (produkty). To znamená, jaké produkty a dokumenty se v dané činnosti vytvářejí nebo upravují, případně v členění:
    - klíčové dokumenty,
    - ostatní produkty,
- zúčastněné profese a odpovědnost (profese, které se účastní prací na dané činnosti a za co odpovídají).
- doporučené techniky a nástroje. Techniky, které se v činnosti používají, jejich možná podpora a ostatní nástroje vývoje IS.

Z výše uvedeného tedy vyplývá, že životní cyklus může být chápán různě. Přesto existují národní a mezinárodní standardy, které popisují posloupnost kroků a přesné určení jaké metody, techniky, nástroje, dokumenty, řízení atd. použít při vývoji IS.

---

[30] Řepa, V. Analýza a návrh informačních systémů. Ekopress, Praha 1999, ISBN: 80-86119-13-0. str. 17-19

7.3. Český standard životního cyklu

V České republice byl v prosinci roku 2002 vydán Standard ISVS 005/02.1[31] pro náležitosti životního cyklu informačního systému, který vydal Úřad pro veřejné informační systémy v Praze. Standard se vztahuje na IS veřejné správy a na projekty akvizice, vývoje, provozu a údržby těchto systémů a věnuje se především dokumentaci a jednotlivým krokům, které musí být provedeny při vývoji IS.

Standard definuje základní postupy a náležitosti procesů životního cyklu informačního systému nebo jeho části s hlavními cíli[32]:

- zajistit řízení a zejména kvalitní organizaci a kontrolu projektů rozvoje IS v rámci organizace správce ISVS,
- zajistit kvalitní řízení vývoje, provozu a údržby informačního systému jako celku,
- připravit věcný podklad pro atestace informačních systémů nebo projektů jejich rozvoje,
- poskytnout správcům informačních systémů veřejné správy možnost efektivněji kontrolovat dodavatele komplexních řešení (zejména externí subjekty) rozšířením počtu a přesnou definicí kontrolních bodů, ve kterých lze zpracovávaný projekt ovlivnit nebo v krajním případě zastavit tak, aby nedocházelo k dalším zbytečným finančním a časovým ztrátám,
- vést jednotnou strukturovanou dokumentaci jednotlivých projektů tak, aby nebyly ohroženy personálními změnami v řešitelských týmech.

Dále jsou standardem sledovány cíle:

- definovat návaznosti jednotlivých projektů v organizaci správce ISVS,
- vést evidenci o jednotlivých informačních systémech, projektech jejich rozvoje a jejich meziresortních návaznostech pro potřeby (a podle metodických požadavků) Úřadu pro veřejné informační systémy,
- zajistit základní kontrolu účelnosti vynakládání finančních prostředků na projekty ICT ze státního rozpočtu.

---

[31] Standard ISVS 005/02.01 pro náležitosti životního cyklu informačního systému [online]. Úřad pro veřejné informační systémy 2002. Dostupné na: http://www.mvcr.cz/micr/files/457/uvis_s005.02.01_v2002c5_20021218.pdf

[32] Standard ISVS 005/02.01 pro náležitosti životního cyklu informačního systému [online]. Úřad pro veřejné informační systémy 2002. Dostupné na: http://www.mvcr.cz/micr/files/457/uvis_s005.02.01_v2002c5_20021218.pdf. str. 4

7.4. Mezinárodní standardy

Mezinárodní standardy pronikají také na území České republiky a jejich dodržování se stává jedinou z možností jak obstát na konkurenčním trhu. V této podkapitole se zaměříme na ty nejznámější.

7.4.1. ISO/IEC 12207

ISO/IEC 12207[33] je mezinárodní standard určený pro softwarové procesy v rámci životního cyklu (Software Life Cycle Processes). Tento standard byl poprvé navržen v roce 1988 a publikován v roce 1995 a stanoven jako společný mezinárodního rámec pro vývoj, dodávky, podporu a údržbu softwaru. Standard se věnuje především třem zásadním procesům: procesům primárního životního cyklu, podpoře procesů životního cyklu a organizačním procesům životního cyklu.

7.4.2. ISO/IEC 15504

ISO/IEC 15504[34] Software Process Improvement and Capability Determination, známý pod zkratkou SPICE, je mezinárodní rámec pro hodnocení procesů, který byl vyvinut Joint Technical Subcomittee v období mezi standardem ISO (International Organization for Standardization) a IEC (International Electrotechnical Comission). Byl odvozen ze standardu procesu životního cyklu ISO 12207 a modely dospělosti (maturity models) jako je model CMM (o modelu CMM je pojednáno v kapitole 8.5.2.1), Bootstrap a Trillium.

Standard obsahuje referenční model, který definuje dimenze procesů a dimenze "způsobilosti" (capability). Dimenze procesů jsou rozděleny do pěti kategorií procesů na dimenzi zákazník/dodavatel, inženýrství, podpora, řízení a organizace.

Vyspělost procesů (capability of processes) je měřena pomocí atributů procesu a jsou definovány mezinárodním standardem devíti atributy jako je výkonnost procesů, řízení procesů, dále definice, rozmístění, měření, kontrola, inovace a optimalizace procesu. Každý z těchto atributů je hodnocen

---

[33] ISO/IEC 12207 Software Life Cycle Processes [online]. SEPT Supplying Software Engineering Standards Information to the World 2008. Dostupné na: http://www.12207.com/

[34] SO/IEC 15504 Software Process Improvement and Capability Determination. Information technology Process assessment Part 1: Concepts and vocabulary, Information technology Process assessment Part 2: Performing an Assessment, Information technology Process assessment Part 3: Guidance on performing an assessment, Information technology Process assessment Part 4: Guidance on use for process improvement and process capability determination, Information technology Process Assessment Part 5: An exemplar Process Assessment Model, Information technology Process assessment Part 6: An exemplar system life cycle Process Assessment Model, Information technology Process assessment Part 7: Assessment of Organizational Maturity [online], Dostupné na: http://www.iso.org/iso/iso_catalogue.htm

pětibodovým hodnocením od nedosaženo (not achieved) až po plně dosaženo (fully achieved).

Standard poskytuje návod na to, jak provádět hodnocení procesů, obsahuje model pro toto hodnocení a doporučené nástroje používané pro hodnocení.

7.5. Závěr

Životní cyklus kompletně pokrývá celý vývoj IS, který se skládá z jednotlivých etap/fází, které na sebe navazují. Každý IS lze popsat pomocí životního cyklu, jehož základními etapami je zachycení požadavků na systém, tvorba konceptuálního a implementačního modelu, implementace, zavedení, testování, udržování systému a jeho provoz a v některých případech také i stažení systému z užívání.

Jednotliví vývojáři si životní cyklus přizpůsobují podle svých vlastních potřeb, tzn. ne vždy se shodují názvy etap a jejich rozdělení do dílčích činností a úkolů u všech projektů. To umožňuje každému přizpůsobit si životní cyklus podle svých potřeb, čímž dochází ke zvýšení efektivity a produktivity.

Pro snadnější pochopení životního cyklu existují národní a nadnárodní standardy. Zmínili jsme český standard ISVS 005/02 z roku 2002, který se vztahuje na IS veřejné správy a na projekty akvizice, vývoje, provozu a údržby těchto systémů, dále se věnuje především dokumentaci a jednotlivým krokům, které musí být provedeny pří vývoji IS.

Nejznámějšími mezinárodními standardy je Standard ISO/IEC 12207 a ISO/IEC 15504. ISO/IEC 12207 je Standard pro softwarové procesy v rámci životního cyklu (Software Life Cycle Processes) a je návodem, jak správně řídit projekt IS. Standard ISO/IEC 15504 (Software Process Improvement and Capability Determination) nám může pomoci při zlepšování procesů v rámci vývoje IS, které vedou k zvýšení efektivnosti a produktivity práce.

8. Metodiky pro vývoj SW

8.1. Úvod

Nyní navážeme na předchozí kapitolu, ve které jsme si definovali základní pojmy a budeme se podrobněji zabývat metodikami. Vysvětlíme si, proč jsou metodiky tak důležité, jak je členíme a jaké jsou vybrané předměty jejich zájmů. Dále se budeme zabývat tradičními metodikami a jejich nástupci, agilními metodikami, které v současnosti nabývají velkého významu.

8.2. Význam metodik

Nejprve si znovu připomeňme, co znamená výraz metodika a co je jejím cílem.

Podle České společnosti pro systémovou integraci (ČSSI) je metodika vývoje IS/ICT *"doporučený souhrn fází, etap, přístupů, zásad, postupů, pravidel, dokumentů, řízení, metod, technik a nástrojů pro tvůrce informačních systémů, který pokrývá celý životní cyklus informačního systému. Určuje kdo, kdy, co a proč má dělat během vývoje IS."*[35] Cílem metodik je redukovat potenciální chyby a problémy v rámci vývoje IS.

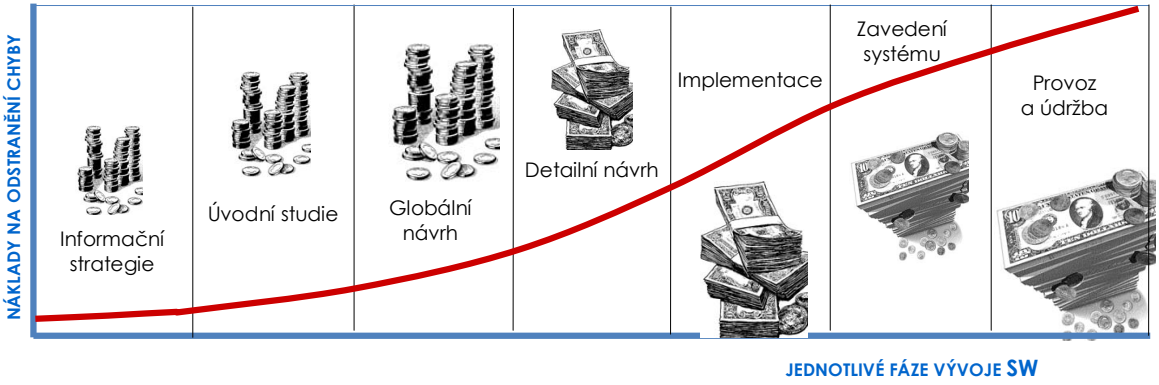
V minulosti měly metodiky jediný cíl dodat IS dle stanovených požadavků. V současnosti se klade na metodiky důraz především z hlediska zvýšení efektivity a získávání nových požadavků, které jsou okamžitě zakomponovány do systému.

8.2.1. Proč využíváme metodik

Výhodou metodik je jejich schopnost odhalit včas chyby a okamžitě je napravit. Pokud by toto nebylo možné, docházelo by k opravě chyb až v pozdějším stádiu tvorby projektu, což by přinášelo zvýšené náklady nejen finanční ale také časové, což u rozsáhlých robustních systémů vede v lepším případě k nepředstavitelnému zatížení nepředstavitelné zatížení?, které může vyústit v obrovská zpoždění a nespokojenost zákazníka, v horším případě je často nemožné zjištěnou chybu opravit. Následující obrázek výstižně zobrazuje rostoucí náklady na odstranění chyby v rámci jednotlivých fází vývoje.

[35] Terminologie. Česká společnost pro systémovou integraci [online]., Dostupné na: http://www.cssi.cz/cssi/terminologie

Obr. 5: Fáze tvorby programového systému, ve které je zjištěna chyba[36]



Jak již bylo uvedeno podkapitole 6.2.2, metodika nám nabízí určitý postup, pomocí kterého můžeme dosáhnout zefektivnění celého procesu tvorby IS. Metodika tedy může pokrývat celý životní cyklus vývoje IS nebo může být jen na specifickou část vývoje. Důležité je, že metodika (příp. metodiky), zahrnuje posloupnost činností včetně určení rolí, zodpovědností vedoucí k úspěšnému vývoji a dokončení IS.

Obrázek Obr. 6: Vybrané předměty zájmu metodik pro tvorbu IS na straně č.34 nám ilustruje, jakým oblastem se zejména věnuje velká pozornost v oblasti metodik. Důvod existence tak nepřeberného množství metodik a CASE nástrojů si můžeme ukázat na jednoduchém příkladu. Představme si, že potřebujeme poslat dopis. Jako nástroje můžeme zvolit tužku a papír, psací stroj nebo počítač. Musím přiznat, že je velmi příjemné si v dnešní době vzít tužku a papír a napsat tímto způsobem dopis (a v poslední době i velmi vzácné), ale přiznejme si, že mnohem efektivnější je napsat tento dopis na počítači a téměř okamžitě ho doručit prostřednictvím emailu protistraně. Abychom neopomenuli psací stroj, je sice mnohem efektivnější než tužka a papír, ale nedosahuje takových možností jako počítač (zejména možnost nepřeberných úprav a možnost odeslání přes Internet). Cíl metody je tedy usnadnit nám výběr, který z nástrojů (tužku a papír, psací stroj či počítač) při tvorbě IS použít. Stejně jako v uvedeném příkladě, si při tvorbě IS můžeme pomocí metodiky vybrat náš specifický nástroj, který bude plně odpovídat našim předpokladům, bude plně vyhovovat naši představě, našim znalostem, zkušenostem či intuici.

Určitá metodika nám tedy může např. poradit, zda si pro psaní dopisu vybrat ten či onen nástroj, ale navíc nám (ta samá či jiná metodika) může pomoci tento nástroj používat. Například návod, jakým postupovat při psaní dopisu, jak dodržovat styly, jakým způsobem provádět úpravy a jak používat elektronickou poštu. Opodstatnění využívání CASE nástrojů založených na metodikách je v tom, že metodiky nám usnadní vývoj a dovedou ho

[36] Řepa, V. Analýza a návrh informačních systémů. Ekopress, Praha 1999, ISBN: 80-86119-13-0. str. 150

do úspěšného konce, ale s nesprávnými nástroji můžeme mít problém daný systém vůbec dokončit. Metodika, ať již zabudovaná v CASE nástrojích či sama o sobě, může být velmi užitečná, neboť vede k vyšší produktivitě vzhledem k lepší komunikaci týmů, umožní dosáhnout větší přizpůsobivost a pružnost vývoje IS. Změny a aktualizace nejsou problémem.

Výše uvedené neplatí obecně, vždy je třeba zohlednit daný projekt a ten uzpůsobit metodice či metodiku projektu (obvykle dochází k přizpůsobení obojího: metodika má určité mantinely, kterým se přizpůsobuje projekt ale na druhou stranu nelze projekt přizpůsobit precizně do nejmenších podrobností metodice – týká se metodik specifických).

Pro použití metodik je klíčová fáze analýzy a návrhu, neboť se zjistilo, že tato fáze vývoje se řadí mezi nejdůležitější a závisí na nich celková úspěšnost projektu. Použití metodik se ukázalo jako velmi prospěšné a užitečné a jejich pozitiva byly ještě znásobeny zavedením CASE nástrojů. Díky úspěšnosti metodik byly vytvořeny CASE nástroje, které z jejich pozitiv vytváří synergický efekt, protože spojují best practices spolu s celkovým pohledem na vybranou část či na všechny dimenze životní fáze vývoje IS.

8.3. Kategorizace metodik

Metodikám, především jejich kategorizaci, agilním metodikám a vzorům pro vlastní návrh metodiky, se věnuje ucelená publikace Buchalcevové[37].

Vzhledem k tomu, že[38]:

1) existují různé technologie vyžadující různé techniky a metody,
2) organizace se liší firemní kulturou,,
3) každý jedinec je jedinečný,
4) každý tým je jedinečný,
5) projekty se liší velikostí týmu,
6) projekty se liší svou důležitostí,
7) projekty se liší podle postavení produktu na trhu,
8) projekt existuje v rámci určitého specifického vnějšího prostředí,

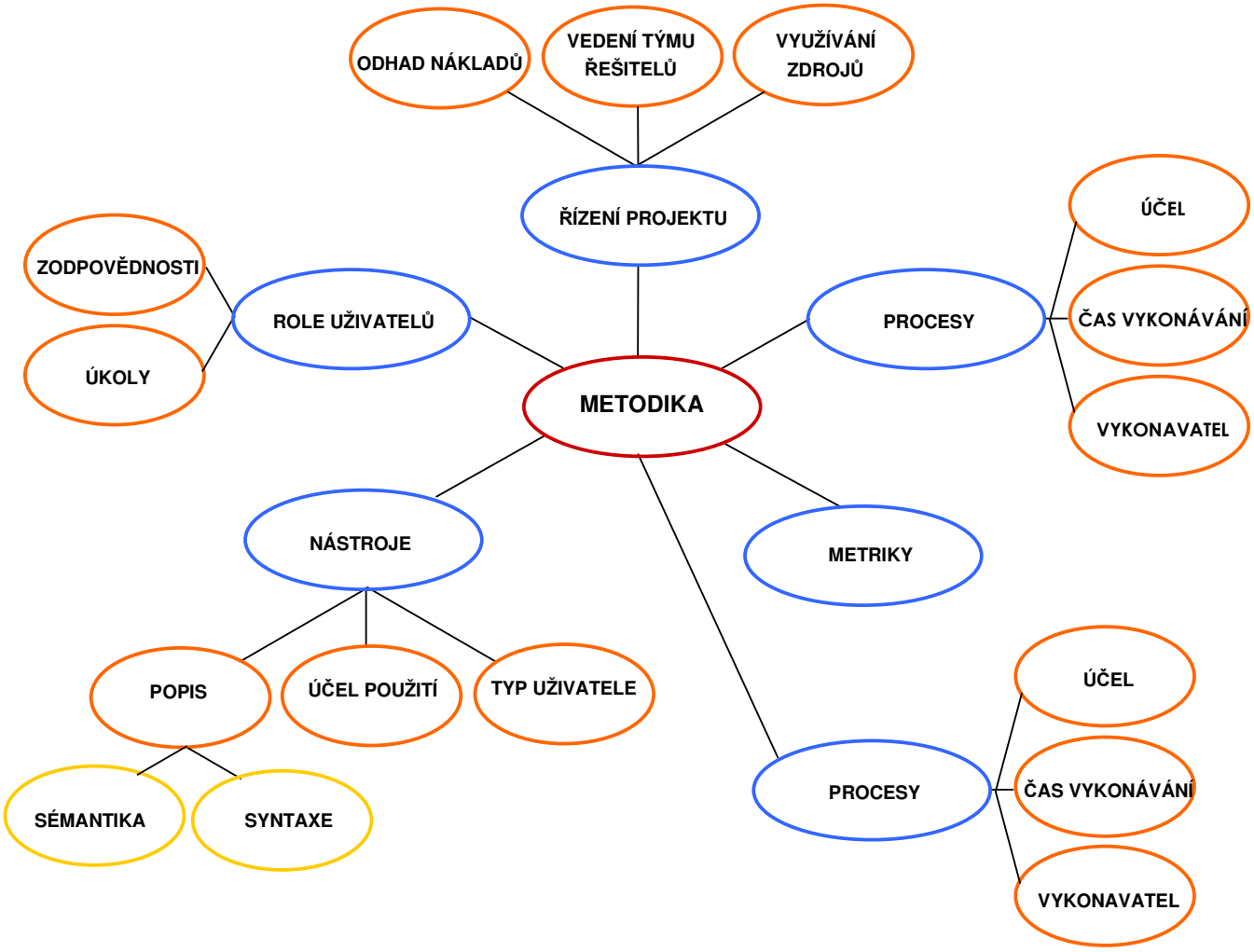
existuje také nepřeberné množství metodik, které pokrývají jednotlivé kombinace nejen výše uvedených skutečností. Rozlišujeme několik kritérií, podle kterých jsou metodiky členěny a my se v následujícím textu budeme věnovat těm nejzásadnějším.

---

[37] Buchalcevová, A. Metodiky vývoje a údržby informačních systémů. Grada, Praha 2005, ISBN 80-247-1075-7

[38] Buchalcevová, A.: Metodiky vývoje programových systémů, In: sborník prací účastníků vědeckého semináře doktorského studia FIS VŠE v Praze, 2003, ISBN 80-245-0518-5

Obr. 6: Vybrané předměty zájmu metodik pro tvorbu IS[39]



8.4. Metodiky z hlediska zaměření

Dle zaměření lze metodiky členit na obecné a speciální.

8.4.1. Obecné metodiky

Jak již název napovídá, obecné metodiky chápou proces vývoje velmi obecně a je třeba je přizpůsobit podle podmínek dané firmě a typu projektu. Příkladem obecné metodiky je metodika RUP a EUP o které bude pojednáno v kapitole 8.5.3.

[39] Smil, P. Úvod do tvorby IS s použitím objektového přístupu. Softwarové noviny, prosinec 1999, cis.12, ISSN: 1210-8472

8.4.2. Speciální metodiky

Speciální metodiky neboli konkrétní metodiky jsou metodiky, které si firmy zhotovují sami. Tyto metodiky jsou přizpůsobeny na míru přímo určité firmě a jsou určeny pro určitý typ projektu pro určitou situaci a další přesně definované podmínky. Proto je velmi těžké a často i nemožné tuto metodiku přizpůsobit jiné firmě. Příkladem speciální metodiky je metodika RUP, která i přes svou „speciálnost" odpovídá široce požadavkům vývoje IS.O této metodice blíže pojednává kapitola 8.5.3.

8.5. Tradiční a agilní metodiky

Další možnost dělení metodik je na tradiční a agilní.

8.5.1. Tradiční metodiky

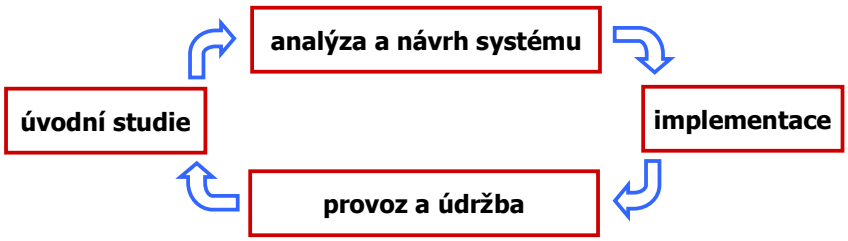
Tradiční metodiky jsou jak historickými „tradičními" přístupy používané v metodikách. (vodopád) či současně době používané a aktuální metodiky (spirála).

Tradiční metodiky jsou založené na sekvenčním modelu, jedná se o jakoukoliv metodika, ve které se nachází určité fáze vývoje, ve kterých se v průběhu vývoje IS pracuje s určitou formální dokumentací, kterou si můžeme představit pod různými diagramy, tabulkami a seznamy. Tradiční metodiky jsou uváděny jako příliš složité, málo čitelné a účinné a proto vyžadují velké množství dokumentace, což komplikuje a oddaluje výslednou implementaci. Tyto metodiky jsou navíc údajně neúčinné v podmínkách nejen rychle se měnících požadavků ale i rychle se měnících technologií.[40]

8.5.1.1. Klasická metoda

Klasický přístup k životnímu cyklu[41] je velice jednoduchý a je tvořen čtyřmi etapami: úvodní studie, analýza a návrh systému, implementace a provoz a údržba.

Obr. 7: Klasická metoda



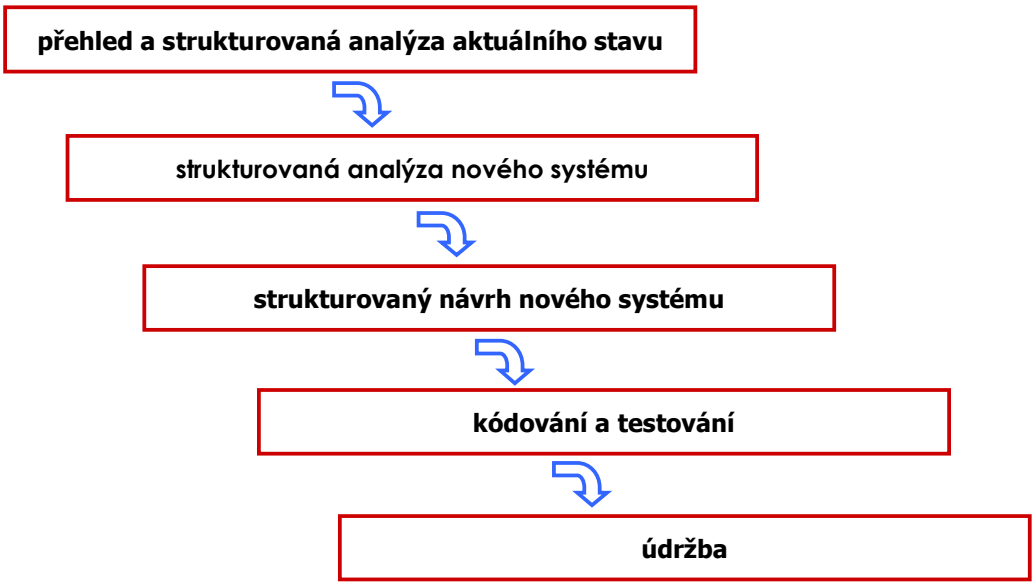
[40] Merunka, V. Metody tvorby informačních systémů [online]. PEF ČZU Praha 2008. Dostupné na: http://kii.pef.czu.cz/~merunka/documents/for_students/U3V/U3V%20-%20metody%20tvorby%20IS%20-%20text.doc. str. 1

[41] Vodáček, L. Rosický, A. Pojetí, poslání a význam informačního managementu. Management Press, Praha 1997, ISBN 80-85943-35-2

8.5.1.2. Yourdonova metoda[42]

Yourdon používá pro popis životního cyklu tzv. „fázový model.“ Jedná se o všeobecnou metodiku, která všeobecně poskytuje osvědčené best practices pro jednotlivé etapy, které jsou zřejmé z uvedeného obrázku.

Obr. 8: Yourdonova metoda



8.5.1.3. Code and Fix Model

Code and Fix Model[43] je velmi často používán v metodologiích softwarového inženýrství. Jeho průběh si můžeme popsat tak, že před samotným vývojem se vytvoří plán, ale není to podmínkou, tento model umožňuje vývoj i bez jakéhokoliv plánování. Dále začíná fáze vývoje a zjišťování problémů. Problémy se zjišťují a řeší až v situaci, kdy se prvně objeví. Vývoj a odstraňování chyb probíhá tak dlouho, dokud není vyvinutý software v požadované kvalitě.

Tento způsob vývoje (pokud ho tak můžeme nazvat, neboť se v podstatě nejedná o žádnou metodiku, protože neexistují žádné postupy, doporučení ani best practices) mohl být schůdným řešením v minulosti, kdy požadavky IS nenarážely na problémy s robustností IS, možnostmi implementace a integrace.

8.5.1.4. Metoda tunel

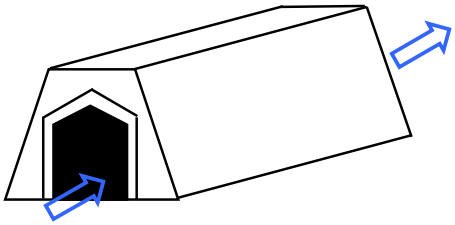
Metoda tunel je jednou z podkategorií metody Code and Fix Model.

---

[42] The Yourdon (Ward-Mellor) Structured Method [online]. Schools of Electronic Engineering and Computer Science. Dostupn0 na: http://www.informatics.bangor.ac.uk/~dewi/modules/rts/YSM_slides.pdf

[43] Lifecycle Models. National Instruments [online]. Dostupné na: http://zone.ni.com/reference/en-XX/help/371361A-01/lvdevconcepts/lifecycle_models/

Obr. 9: Metoda řízení projektu tunel



Jedná se o postup tvorby IS[44], ale ani toto není příhodným vysvětlením, neboť vlastně žádné postupy neobsahuje. Tento přístup můžeme tedy nazvat "antimetodou" a není doporučeno ho používat.

V první fázi vývoji, raději řekněme, že na počátku projektu, (neboť, jak již bylo zmíněno, žádné fáze nejsou definovány), vkročíme do černého tunelu, kterým poslepu postupujeme dál. Našim cílem je jím úspěšně projít, tedy najít cestu ven (vyvinout IS). Problém je, že v tunelu nic nevidíme a proto narážíme od zdi ke zdi. Cílem vedoucího projektu je řídit "náš průchod tunelem" tak, abychom jím co nejrychleji prošli (přičemž ani sám vedoucí neví, kudy vede cesta ven) a zbytečně si cestu nekřižovali a nenaráželi neustále do stěn či do sebe navzájem. Jde tedy o jakési ad hoc řešení bez předem naplánovaného a systematicky řízeného postupu.

Takto řízené projekty mají obrovské problémy s dokončením (odevzdá se zpožděním nebo s omezenou funkčností či s překročením rozpočtu atd.) a často se stává, že ani dokončeny nejsou (vzdálené světlo, které nám ukazuje konečně směr, jakým se vydat z temnot je protijedoucí vlak).

Chaotický vývoj ve firmě vede k hektické práci přesčas, shonu a nervozitě, která přispívá i ke špatným vztahům na pracovišti. Software se neustálé předělává stále dokola.

Nevýhody metody tunel jsou z výše uvedeného zřejmé. Jde především o:

- vysoké náklady jak finanční, tak na náročnost a řízení,
- velmi často zpoždění či nedodání systému,
- funkce systému se neshodují s požadavky zákazníka,
- metoda nevyužívá best practices, metodiky ani postupy procesů, které se osvědčily,
- intuitivní řízení bez návodů a příruček záleží na zkušenostech vedoucího projektu (dva stejné projekty pod vedením identického vedoucího neprobíhají stejně, výstupy se liší),
- náročnost práce pro vedoucího projektu
- nelze nic předpokládat, provádět odhady, nikdo neví, co ho čeká,

---

[44] Kraval, I. Návrh informačních systémů pomocí UML, OOP a vzorů [online]. Server objektových technologií 2006. Dostupné na: http://www.objects.cz/clanky/clanky_IS/NavrhIS.pdf, str. 2-5

- nedá se zavést požadavek na opětovnou použitelnost, čímž vzniká při dalším vývoji zbytečná práce,
- velmi těžké a někdy i nemožné najít chyby (o některých se ani neví),
- nízká transparentnost projektu,
- nepřehledná a zbytečně složitá architektura,
- opakovaná práce způsobuje redundanci v jednotlivých agendách (osoba je v systému evidována několikrát, nutné zavedení nové agendy),
- při malé změně v systému hrozí obrovský problém zprovoznit systém,
- nutnost oprav chyb i po implementaci (objevování chyb i po odevzdání odběrateli),
- nízká kvalita systému.

Vzhledem k výše uvedeným zásadním problémům metody tunel není doporučeno její používání.

8.5.1.5. Byrokratická metoda

Byrokratická metoda je jakousi snahou o přechod z metody tunel na jakoukoliv řízenou metodu. Je to ale slepá cesta, kterou mnohdy dospějeme ještě k horším výsledkům a je lepší ji vůbec nepoužívat.

Byrokratická metoda[45] je uváděna jako ještě horší a nebezpečnější metoda v porovnání s metodou tunel. Postup začíná stejně jako v metodě tunel, ale časem začne být patrné, že je něco špatně a že je třeba přejít k jinému systému řízení. Vzniká tedy potřeba zavedení postupů a metodik řízení. Problém je v tom, že místo, aby firma zavedla ověřené best practices, užívané v jiných firmách, nechá tyto postupy a metodiky napsat své zaměstnance, kteří jim zpravidla příliš nerozumí

Byrokratickou metodu můžeme charakterizovat takto:

- snaha zavést metodiky a postupy většinou bez jakýchkoliv zkušeností a bez dodržování základních pravidel pro vývoj IS,
- metodiky a postupy se vymýšlejí a nejsou obvykle ověřeny v praxi,
- metodiky a postupy jsou uznávané jako konečné pravdy, musí se dodržovat a nelze je měnit,
- zavádění byrokratické metodiky zpomaluje vývoj IS, což přispívá k nespokojenosti zákazníka,
- byrokratická metoda zpravidla končí neúspěchem, neboť:

  1) zaměstnanci dodržují nesmyslné metodiky a postupy, což vede k totálnímu krachu projektu (horší případ),

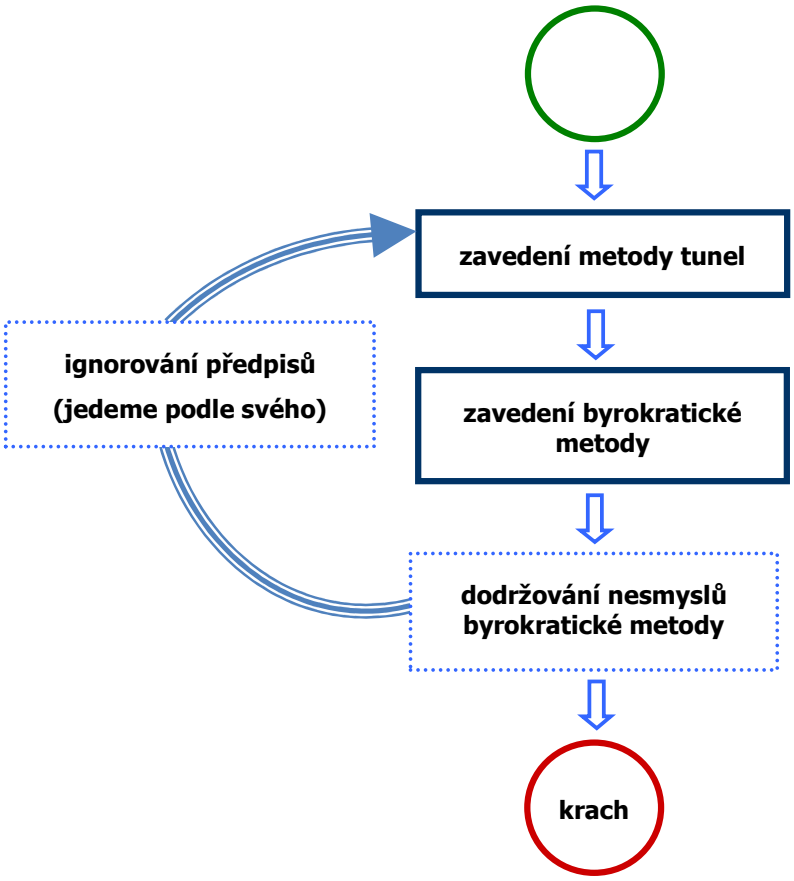
  2) se od zavádění metody ustoupí a následuje návrat k metodě tunel, kdy se projekt jen tak tak dokončí s jakousi funkcionalitou (lepší případ).

---

[45] Kraval, I. Návrh informačních systémů pomocí UML, OOP a vzorů [online]. Server objektových technologií 2006. Dostupné na: http://www.objects.cz/clanky/clanky_IS/NavrhIS.pdf, str. 6 - 8

Proto zde platí stejné doporučení jako u metody tunel, tedy raději ji vůbec nepoužívat.

Obr. 10: Byrokratická metoda[46]



8.5.1.6. Stagewise model

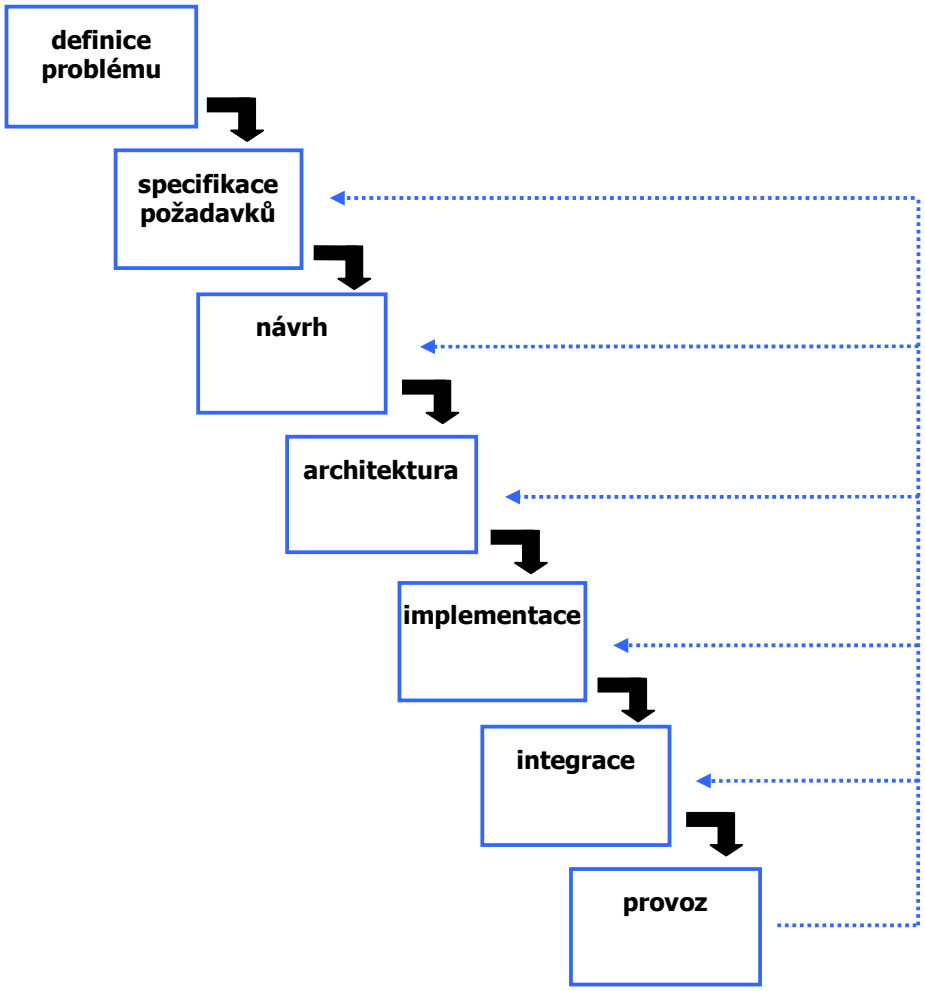
Stagewise model[47] vznikl jako reakce na potřeby řídit vyvíjený software dlouhodobě, což do té doby bylo velice problematické. Jedná se o model, který je založený na přesném dodržování jednotlivých, na sebe navazujících fází životního cyklu při vývoji IS, kterými jsou: definice problému, specifikace požadavků, architektura IS, návrh IS, implementace, integrace a provoz. Problémem tohoto modelu je, že ke kontrole jednotlivých fází a tudíž i k opravě vzniklých chyb (příp. úpravám IS podle nově vzniklých požadavků) dochází až ve fázi, kdy je projekt dokončen, což přináší obrovské náklady na tyto úpravy

[46] Kraval, I. Návrh informačních systémů pomocí UML, OOP a vzorů [online]. Server objektových technologií 2006. Dostupné na: http://www.objects.cz/clanky/clanky_IS/NavrhIS.pdf, str. 8

[47] Likelihood methods for testing group problem solving models with censored data. Psychometrika 1978, vol. 43, str. 353 - 366

viz. Obr. 5: Fáze tvorby programového systému, ve které je zjištěna chyba na straně 19.

Obr. 11: Stagewise model[48]



8.5.1.7. Metoda vodopád / fontána

Následovník metody stagewise byla metoda vodopád, která převzala z výše zmíněného modelu to nejlepší. Metoda vodopád se velmi podobá metodě stagewise jen s jedním velmi podstatným rozdílem a tím je, že jakoukoliv zjištěnou chybu či odklon od požadavku je možno okamžitě napravit. Problém je ale patrný již z prvního pohledu na obrázek č. 41. Je možné se vrátit pouze do předchozí fáze a tento problém opravit, což znamená sice velké zlepšení z hlediska vývoje oproti metodě stagewise, ale ne z hlediska ostatních pokročilých metod, kterou je např. spirála.

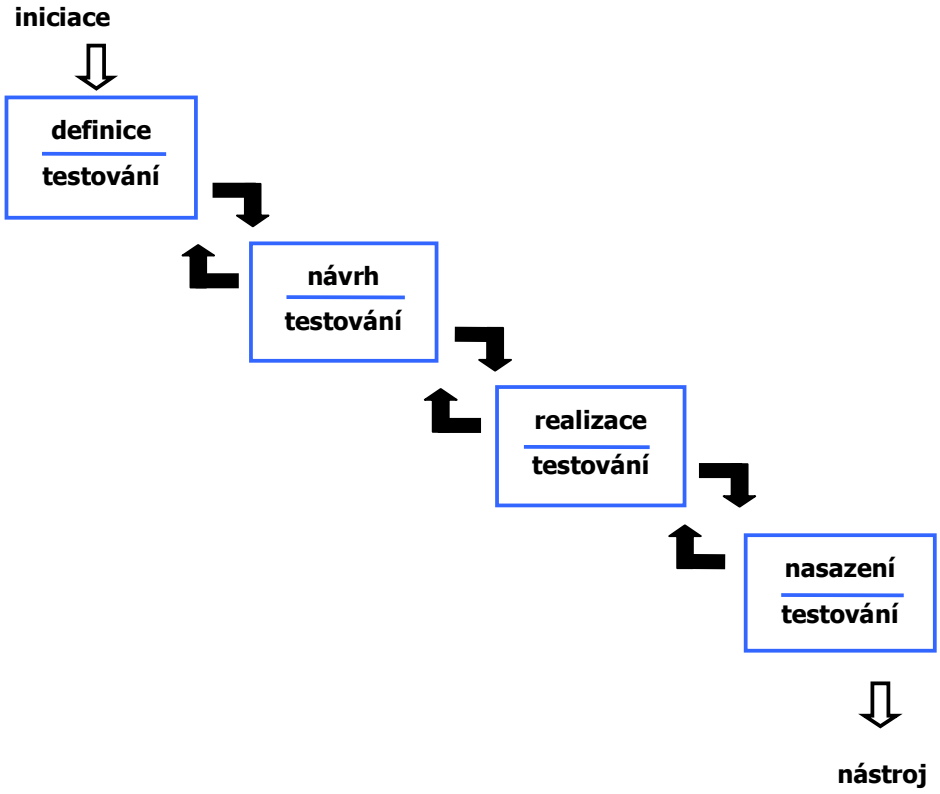
Postata modelu spočívá v tom, že projekt je rozdělen do určitých etap vývoje, které jsou jasně definovány a rozděleny do určitých časových úseků. Prostřednictvím milníků jsou definovány konce jednotlivých etap. Problém

[48] Kadlec, V. Agilní programování: Metodiky efektivního vývoje softwaru. Computer Press, Brno 2004, 1. vyd. ISBN: 80-251-0342-0. str. 34

nastává při potřebě provést podstatnou změnu, neboť výstupy z jednotlivých etap jsou závazné. Pokaždé, kdy nastane ve vývoji milník, vyhodnocuje se provedení předešlé etapy a postupně tak vznikají jednotlivé části, jako např. nejprve dokumenty týkající se analýzy, následují dokumenty týkající se designu a dále dokumenty týkající se programování. Název vodopádová metoda byla odvozena od toho, že dokumenty bývají předávány z jedné fáze do druhé směrem shora dolů.

Obrázek uvedený dole se skládá z několika fází, kterým jsou přiřazeny určité role[49], tedy zadání má na starost pracovník v roli analytika, návrh v roli architekta, realizace se skládá z procesu psaní kódu prováděného rolí programátora a testování prováděného rolí testera, finální fáze nasazení provádí role podpora. Nikdo a nic nevylučuje, aby se jednotlivé role prolínaly či naopak byly striktně oddělené. Tyto role se podílí na průběhu vodopádu a mají určené činnosti a odpovědnosti za jejich provedení.

Obr. 12: Metoda řízení projektu vodopád[50]



Největším negativem této metody je její těžkopádnost a velmi náročné řízení lidských zdrojů. Problém je, že nejprve vznikne kompletně celá analýza. Po

[49] Klobasa, P. Obhajoba vodopádového vývoje [online]. Oxyonline: vývojářský blog 2008. Dostupné na: http://vyvojari.oxyonline.cz/vodopadovy-vyvoj-obhajoba

[50] Nešetřil, V. Fázová organizace projektu a průběhové modely [online]. Vysoká škola báňská - Technická univerzita Ostrava 2001. Dostupné na: http://fmmi10.vsb.cz/639/qmag/mj19-cz.htm#k

analýze následuje kompletní design systému a teprve po jeho dokončení následuje programování kódu. Tato metoda neumožňuje provádění zásadnějších změn při vývoji IS, což je velkým problémem, protože přizpůsobování IS je v dnešní době jedním z nejzákladnějších požadavků na postupy a metodiky při vývoji systému, neboť se málokdy stává, že zákazník nepožaduje změnu po celý průběh vývoje IS. Dále je nutno zmínit, že žijeme v době změn a ne vždy se provádí identický či podobný projekt a stejně tak jako se vyvíjí HW a SW, stoupá i náročnost na změnu stávajících postupů a metodik, tedy i na změny IS.

Vodopádový model je příkladem sekvenčního modelu životního cyklu, který je založený na myšlence, že existuje určitý systematický postup na sebe navazujících činností, které vedou k vytvoření IS. Výhodou je jeho jednoduchost při sledování postupu řešení a řízení, nevýhodou je nutnost ihned na začátku stanovit přesné a úplné požadavky.

Rozšířením vodopádového modelu je tzv. Test Driven Development, což je model, který obsahuje navíc specifikaci, dále se na konci každé fáze provádí testy. Dalším rozšířením je prototypový model (Prototyping Model), který je opět rozšířením vodopádového modelu, tentokrát o vytvoření prototypu, díky kterému dochází ke zpřesňování požadavků.[51]

8.5.1.8. Přírůstková metoda

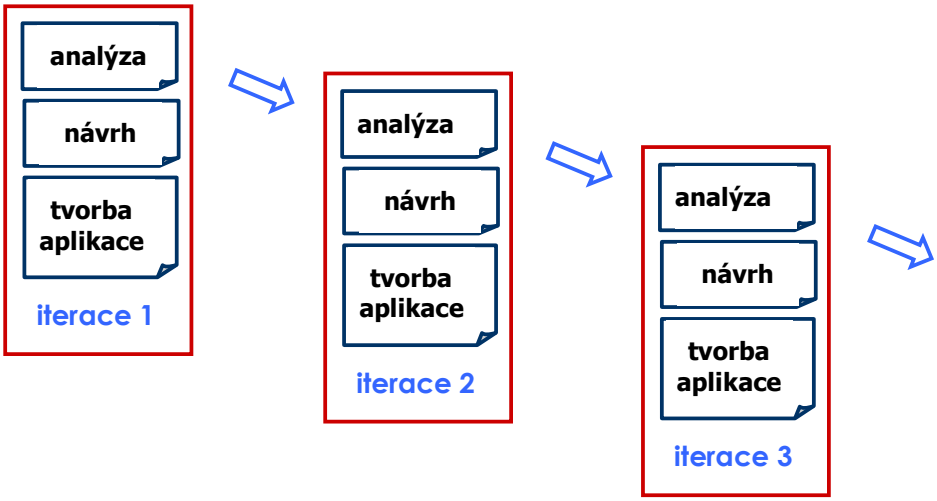
Přírůstková metoda je známá také pod názvem iteračně-interaktivní přístup a je další významnou metodou spolu s dalšími metodami, kterou je např. metoda přírůstková (Incremental Model) neboli evoluční (Evolutionary Model) a spolu s Exploratory Programming a řadí se mezi předchůdce iterativního programování.[52]

---

[51] Kadlec, V. Agilní programování: Metodiky efektivního vývoje softwaru. Computer Press, Brno 2004, 1. vyd. ISBN: 80-251-0342-0. str. 36

[52] Kadlec, V. Agilní programování: Metodiky efektivního vývoje softwaru. Computer Press, Brno 2004, 1. vyd. ISBN: 80-251-0342-0. str. 36

Obr. 13: Přírůstková metoda řízení projektu[53]



"Při iterativním vývoji nejsou přítomny klasické fáze, naopak sběr požadavků, design, implementace, testování a nasazení probíhá průběžně po celou dobu tvorby softwaru."[54] Neexistují tedy žádné pevně stanovené technologické a výrobní procesy.

Podstata této metody spočívá v tom, že vedoucí projektu rozdělí vývoj IS na jednotlivé etapy a tyto etapy na jednotlivé procesy a procesy na jednotlivé úkoly. Celý projekt IS je tedy tvořen jednotlivými menšími projekty, nazývanými přírůstky či iterace, odtud pochází i název metody. Pro tyto, nazvěme je subsystémy, je možné provést samostatný návrh, implementaci i uvedení do provozu. Každý subsystém je tedy samostatný projekt, který prochází všemi fázemi životního cyklu.

Rozdělení do jednotlivých iterací přináší výhody[55], jakými jsou např. Velice rychlé rozpoznání rizik a jejich eliminace a možnost spuštění verze v rámci každé iterace. Výhodou je, že funkcionalita celého systému zůstane za všech okolností zachována. Tato metoda je složením několika vodopádů a probíhá tak, že po dokončení vývoje jedné iterace, tzn. jedné části IS, následuje postupný vývoj po jednotlivých přírůstcích. Metoda je vhodná pro objektově orientované programování a je nutné při její aplikace nezapomenout kromě přidávání tříd, objektů a vazeb také na funkcionalitu objektů a nové vnitřní stavy objektů.

Inkrementální vývoj patří do tzv. evolučního modelu životního cyklu, kdy je systém vyvíjen postupně po menších částech a vzniká podle neustále se měnících požadavků.

---

[53] Arlow, J. UML a unifikovaný proces vývoje aplikací. Computer Press, Brno 2003, ISBN 80-7226-947-X

[54] Technologie - rychlokurz: Iterativní vývoj softwaru, Computerword srpen 2007, ISSN: 1210-9924. str. 27

[55] Informační systémy [online]. Unicorn 2008. Dostupné na: http://www.unicorn.eu/cz/produkty/systems/index.php?id=15630

8.5.1.9. Spirálový model

Nyní se podíváme na model současného vývoje IS. Model vychází především z nedostatků vodopádového modelu a jeho novinkou je iterativní přístup s opakovanou analýzou rizik.

Spirálový model definoval Barry Boehm již v roce 1988 ve článku "Spirálový model pro vývoj a rozšiřování"[56]. Přínos tohoto modelu spočíval v tom, že se jednalo o první model, který vysvětloval, proč záleží na iteracích.

Výhodou modelu je, že umožňuje použití vyvíjeného softwaru již v časných fázích modelu. Rozdělením modelu do jednotlivých kroků se snižuje riziko chyb a tím i náklady na jejich odstraňování, neboť s pokročilostí projektu náklady na odstranění chyb rostou v přímé úměrnosti s dobou vývoje softwaru.

Model je nazýván jako spirálový, neboť jednotlivé kroky se při vývoji systému opakují ve spirále po určitých cyklech. Po ukončení každého cyklu následuje fáze hodnocení a předložení dílčích výsledků komisi. Vždy se ale pokračuje na vyšší úrovni zvládnuté problematiky, a čím se zvyšuje úroveň projektu a čím více se přibližujeme ke středu spirály, tím rostou postupně i náklady vydané na projekt.

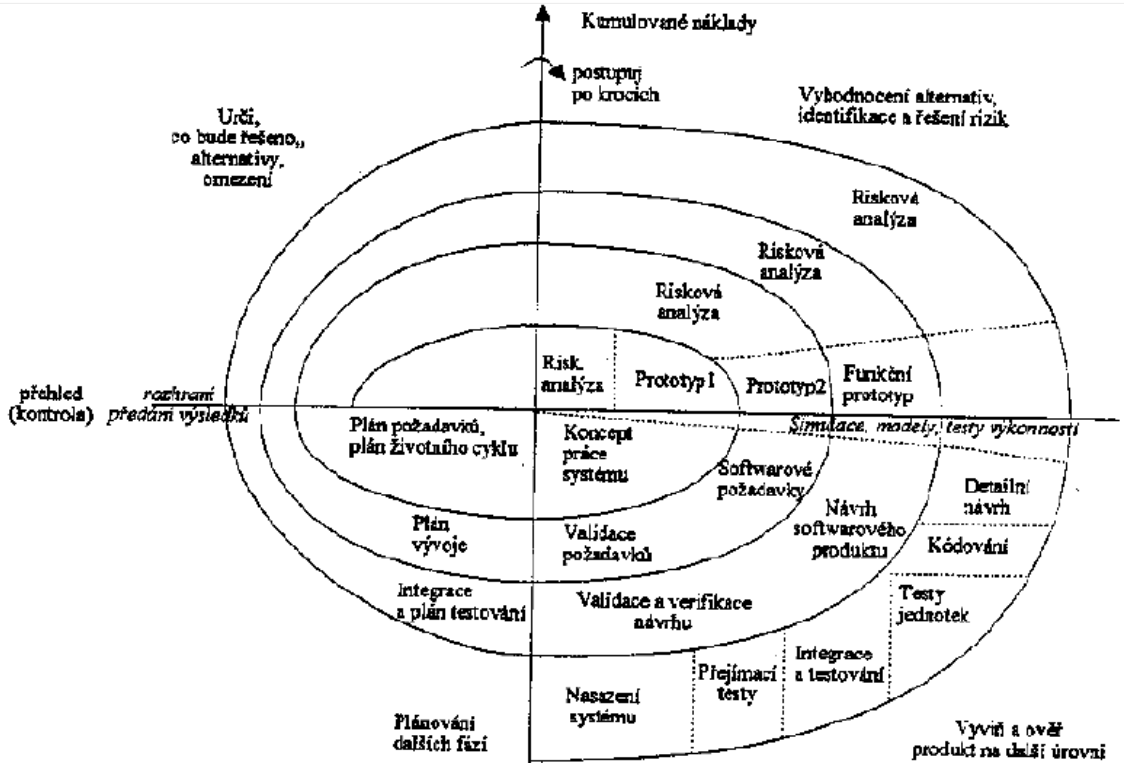
Obrázek uvedený dole je rozdělen do čtyř kvadrantů[57]:

- IV. kvadrant:
    - definice cílů a alternativ vedoucích k dosažení těchto cílů, omezení vyplývající z daných alternativ,
- I. kvadrant:
    - vyhodnocení jednotlivých variant a analýza jejich rizika, zvolení varianty, prototyping,
- II. kvadrant:
    - simulace, testování, modely + ostatní části, které po spirále tvoří v podstatě model vodopád,
- III. kvadrant:
    - kvadrant plánování.

---

[56] Boehm, B. a Spiral Model of Software Development and Enhancement [online]. Computer May 1988. Dostupné na: http://www.ieee.org/portal/site. str. 61 - 72

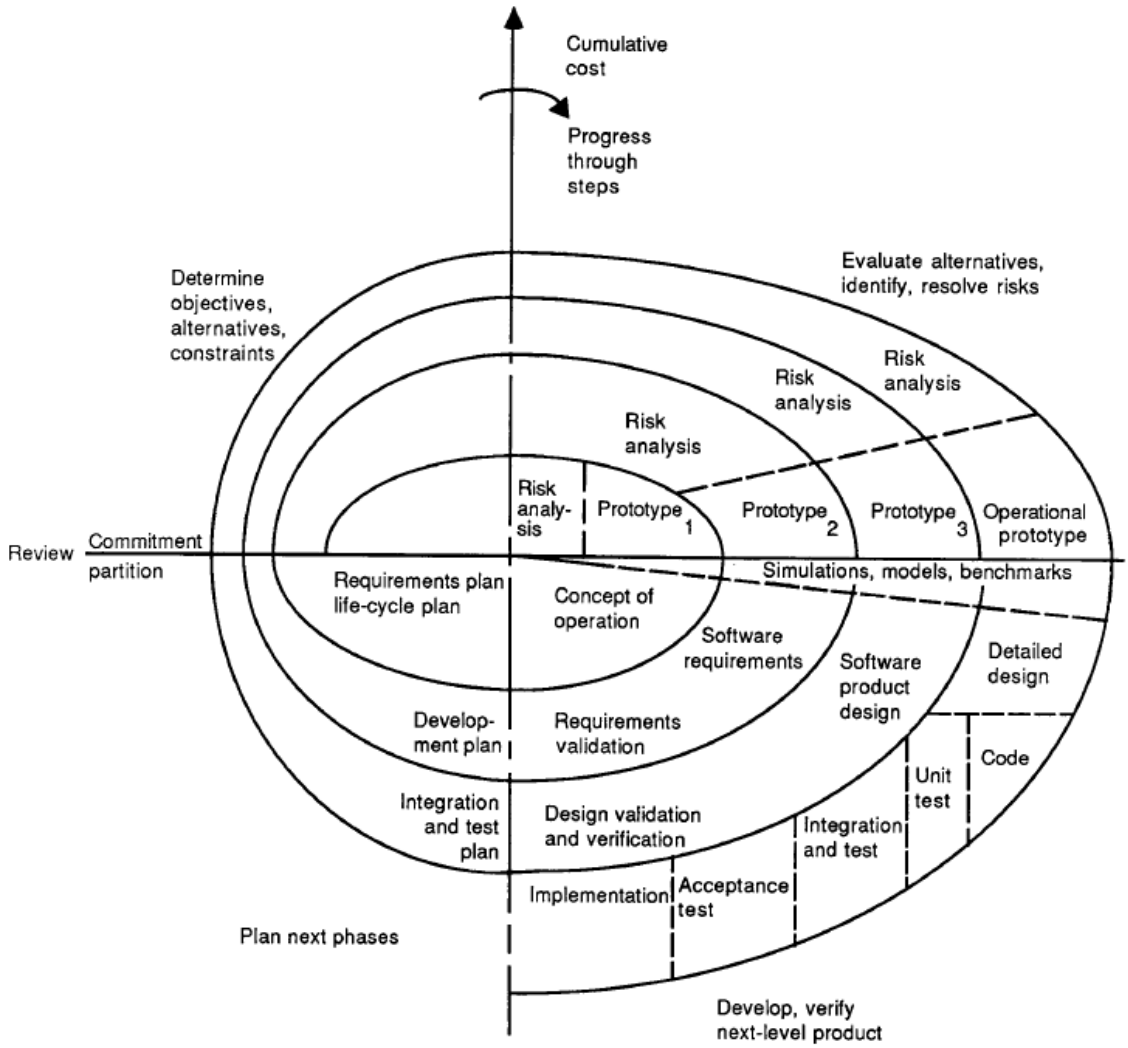
[57] Pavelka, J. Softwarové inženýrství [online]. Matfyz 2008. Dostupné na: http://wiki.matfyz.cz/wiki/SWI026_Pavelka#Model_vodop.C3.A1d



Obr. 14: Spirála[58]

[58] Analýza IS, modely životního cyklu IS [online]. Dostupné na: http://iss.unas.cz/ids/02.pdf. str.4

Obr. 15: Původní spirálový model od Barryho Boehma



Po spirále se postupuje podle dílčích, na sebe navazujících, kroků:

- záměr, celkový plán,
- požadavky, plán vývoje,
- návrh systému, jeho verifikace a validace, plán integrace,
- detailní plán, kódování, integrace, instalace.

Spirálový model se hodí pro větší projekty. Pro malé není vhodný, protože při vstupu do další fáze je zapotřebí stále vyšších nákladů na určení cílů, vyhodnocení, vývoj a následné plánování. Kadlec[59] upozorňuje na nevhodnost spirálového modelu při vývoji databázové aplikace.

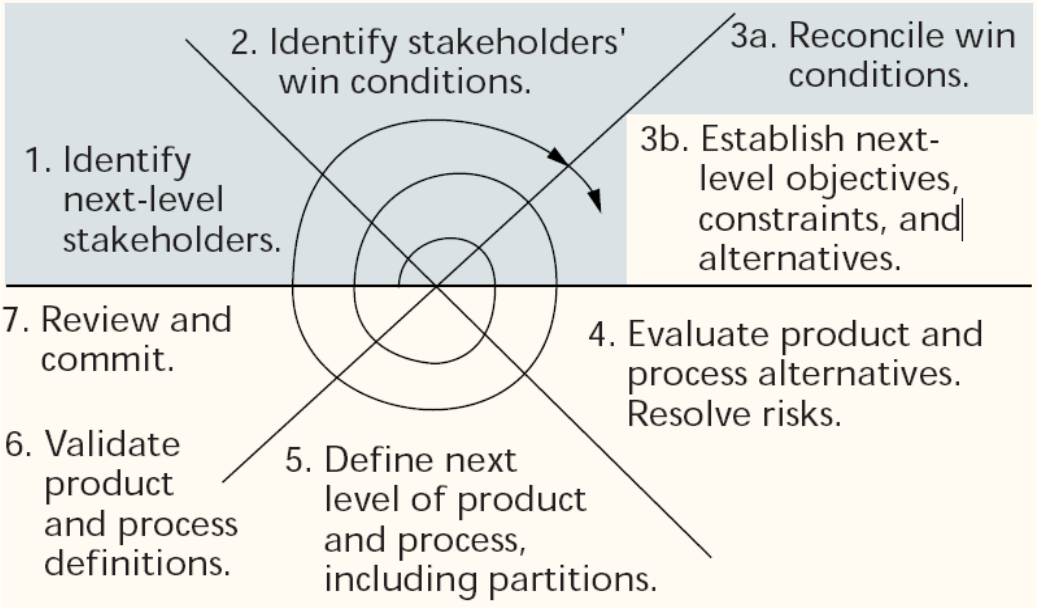
[59] Kadlec, V. Vyvíjíme databázovou aplikaci: 3. díl jak na to [online]. Databázový svět 2003. Dostupné na: http://www.dbsvet.cz/view.php?cisloclanku=2003051203

8.5.1.10. Model Win-Win

Model Win-Win[60] představil Barry Boehm v roce 1994. Vznikl na základě nedostatků spirálového modelu, kterými byla především nutnost definovat na začátku každé otočky spirály cíle, alternativy a omezení s tím, že chybělo uvedení jak toto konkrétně provést.

Každá z fází původního spirálového modelu je v modelu Win-Win podrobněji rozpracována do dalších tří dalších etap, které mají své síle, alternativy a omezení, které vychází z důkladného poznání zákazníka a také z analýzy strategie a rozpracování postupů, které mohou vést k úspěšnému projektu.[61]

Obr. 16: Model Win-WIn[62]



Z obrázku Obr. 16 vyplývá, že model Win-Win se skládá z následujících fází:

1) identifikace účastníků projektu,
2) poznání zákazníka,
3) zjištění podmínek pro úspěch a stanovení cílů pro další fáze,
4) vyhotovení alternativ a provedení analýzy rizik,
5) definování další úrovně procesu a produktu,

---

[60] Boehm, B. Egyed, A. Kwan, J. Port, D. Shah, A. Madachy, R. Using the WinWin Spiral Model: a Case Study, červenec 1998

[61] Kadlec, V. Agilní programování: Metodiky efektivního vývoje softwaru. Computer Press, Brno 2004, 1. vyd. ISBN: 80-251-0342-0. str. 26

[62] Boehm, B. Egyed, A. Kwan, J. Port, D. Shah, A. Madachy, R. Using the WinWin Spiral Model: a Case Study, červenec 1998, str. 34

6) ověření definice procesu a produktu,
7) revize a schvalovací procesy.

8.5.2. Příklady tradičních metodik

V následující podkapitole se budeme zabývat nejvýznamnějšími příklady metodik, používaných pro řízení a podporu softwarových projektů v rámci tradičních metodik. Tyto metodiky se zabývají popisem, plánováním, řízením a měřením procesů. Jsou založeny zejména na vodopádovém, iterativním či inkrementálním vývoji. Obsahují přesně definované postupy a patří do samostatné kategorie metodik pro hodnocení softwarových procesů.[63]

8.5.2.1. SW-CMM: Capability Maturity Model for Software

V posledním desetiletí byla část úsilí zaměřena na pochopení metaprocesu (více se o metaprocesech a metamodelech dočtete v kapitole 10), především organizací Institut pro softwarové inženýrství[64], jejichž velmi dobře známý model zralosti (Capability Maturity Model[65]) definuje pět úrovní zralosti procesu[66] a poskytuje pokyny a pravidla, jak je postupně zlepšit. Model byl vyvinut pro hodnocení dodavatelů softwarových řešení pro ministerstvo obrany Spojených států amerických Institutem pro softwarové inženýrství.

Model SW-CMM definuje pět úrovní zralosti softwarových procesů a na základě jejich definice identifikuje nejdůležitější oblasti pro zlepšení procesů, na které by se měla organizace zaměřit. Definuje oblasti procesů a cíle, které by měly být dosaženy tím, že říká co by organizace měly dělat, neradí však již jak by to měly dělat. CMM se řadí mezi globální metodiky a zaměřuje se

---

[63] Buchalcevová, A. Metodiky budování IS/ICTNávrh metodického rámce IS/ICT: Metodiky budování IS/ICT. KIT VŠE Praha 2004. str. 12

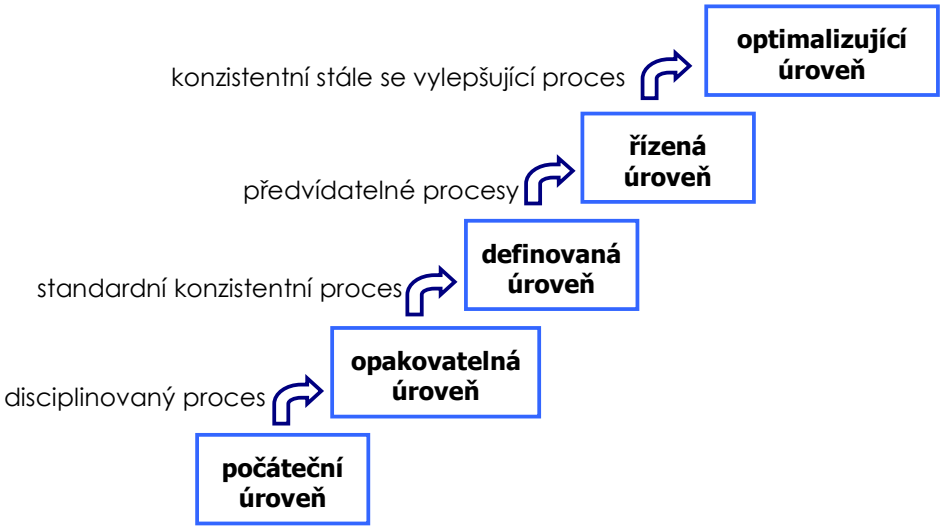
[64] Fuggetta, A. Milano, P. CERFIEL a Classification of CASE Technology [online].The world's leading professional association for the advancement of technology 1993. Dostupné na: http://www.ieee.org/portal/site (navštíveno 5. dubna 2008). str 27

[65] SW-CMM neboli Model zralosti softwaru (Capability Maturity Model for Software) vyvinul Institut pro softwarové inženýrství (SEI - Software Engineering Institute), jež je součástí Carnegie Mellon University. Jeho účelem bylo hodnocení dodavatelů softwarových řešení v roce 1995 pro US Department of Defense. Na jeho základě vzniklo mnoho modelů, jmenujme ty nejvýznamnější: v roce 2000, vznikl na jeho základě integrační model zralosti (CMMI - Capability Maturity Model Integration), dále SECM - Systems Engineering Capability Model a Integrated Product Development Capability Maturity Model (IPD-CMM). Při zavádění těchto modelů vyvstal problém s jejich integrací, což vedlo ke tvorbě integračního modelu CMMI - Capability Maturity Model Integration. Tento model má za úkol spojovat a rozšiřovat modely a podporovat jejich zlepšování, které probíhá dvěmi způsoby: průběžné a postupné. První verze CMMI modelu byla publikována v prosinci roku 2000, v současnosti je k dispozici verze 1.2. Více na http://www.sei.cmu.edu/cmmi/

[66] Pressman, R. S. a Pracittioner´s Approach: Software Engineering. McGraw-Hill, New York, 1992

na veškeré procesy v rámci celé organizace, především na řízení softwarových procesů, méně na softwarově inženýrské postupy.[67]

Obr. 17: Capability maturity model[68],[69]



V počáteční úrovni, označováno jako initial, se dá o procesu mluvit jako o neexistující zralosti, neboť softwarové procesy nelze nijak pozorovat, protože jsou naprosto náhodné. Daná organizace nemá stabilní prostředí pro vývoj a údržbu softwaru. V případě problému reaguje spontánně.

Přechod od počáteční úrovně k opakovatelné (repeatable) probíhá tak, že organizace si postupně uvědomuje, že není dobré řešit problémy "ad hoc" a na individuální bázi, ale že je třeba vzniklé problémy řešit na základě určitých pravidel. To vede ke vzniku opakovatelné úrovně.

V úrovni „opakovatelná" (repeatable) se organizace snaží vytvořit standardní procesy. Jejich využívání je ale jen intuitivní, takže dosti často dochází k duplikaci činností jednotlivými pracovníky. Důvodem duplikace je, že standardizace postrádá přesnější definice, které by umožnili lepší aplikaci.

V definované úrovni (defined) má organizace přesně definované, dokumentované a standardizované procesy, které jsou vzájemně integrovány v celé organizaci. Zaměstnanci jsou vyškoleni pro jejich využívání, ale realizace je opět individuální.

---

[67] Buchalcevová, A. Návrh metodického rámce IS/ICT: Metodiky budování IS/ICT. KIT VŠE 2004. str. 44

[68] Capability Maturity Model V1.1 Pocket Guide. Judy Bamberger. Merant, ltd.. 1999

[69] Buchalcevová, A. Agilní a rigorózní metodiky: Úrovně zralosti CMM. Dostupné na: http://nb.vse.cz/~vorisek/FILES/4IT215_materialy_k_predmetu/MetodikyIT_Buchalcevova.ppt. str. 13

Řízená úroveň (managed) je fáze, ve které má organizace definovány standardy a popisy procesů, existují detailní metriky softwarových procesů i kvality produktu. Softwarový proces je předvídatelný.

Optimalizovaná úroveň (optimizing) je úroveň, kdy jsou procesy v organizaci optimalizovány na základě jejich sledování a zlepšovány pomocí "best practices". Optimalizace a sledování "best practices" jsou činnostmi, zakomponovanými v procesech.

Model zralosti SW je jedním z nejznámějších příkladů hodnocení softwarových procesů[70].

Jednou z nejpoužívanějších metodik je metodika "Rational Unified Process" (RUP)[71] a metodika "Object-Oriented Software Process Pattern"[72]. Další velmi významnou metodikou je "Enterprise Unified Process" (EUP), která je doplněním metodiky RUP a dále metodika OPEN. V praxi se metodika RUP a EUP vzájemně kombinují, obou se tedy využívá při vývoji současně.

8.5.3. Metodika RUP

Rational Unified Process®

Metodiku RUP vyvinula společnost Rational, kterou od ni odkoupila společnost IBM za 2 miliardy amerických dolarů. Vychází z metodiky Objectory Process, která vytvořil již v roce 1987 Ivar Jacobson. Metodika vychází z případů užití a je založena na objektově orientovaném přístupu. Metodiku společnosti Rational používalo několik významných firem, se kterými se společnost sloučila a došlo ke vzniku několika metodik, např. Rational Objectory Process 4.0 po sloučení se švédskou firmou Objectory AB či rozšířená verze Objectory Process 4.1 s firmou SQA a Requistite. Po akvizici se společností Pure Atria vznikla verze Rational Unified Process 5.0.

Metodika je založena na best practices vývoje softwaru, kterými jsou iterativní vývoj, řízení požadavků, použití komponentové architektury, vizuální modelování, kontrola kvality software a řízení změn:

- iterativní způsob vývoje softwaru:
    - iterativnímu způsobu vývoji se budeme věnovat v kapitole 8.5.1.8
- správu požadavků:
    - jedná se o takovou správu požadavků, kdy se veškeré dokumenty sbírají jen a pouze na začátku projektu, předání později kdykoliv během vývoje SW je nepřípustné. Aktivní správa umožňuje

---

[70] [70] Buchalcevová, A. Návrh metodického rámce IS/ICT: Metodiky budování IS/ICT. KIT VŠE 2004. str. 12

[71] Ambler, S. W. Process Patterns: Buiding Large-Scale Systems Using OO Technology, Cambridge University Press, 1998, ISBN 0-521-64568-9

[72] Ambler, S. W. More Process Patterns: Delivering Large-Scale Systems Using OO Technology, Cambridge University Press, 1999, ISBN 0-521-65262-6

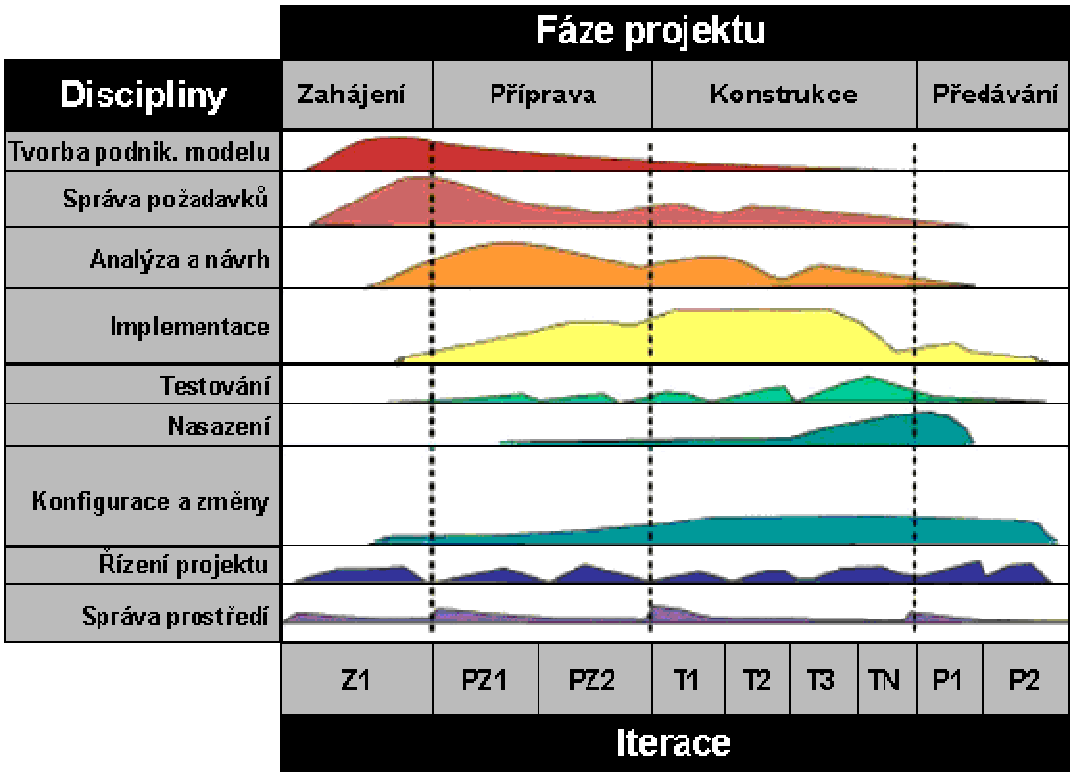
neustálou kontrolu a kontakt vývojové firmy a zákazníka, možnost zpřesňování požadavků.

- architekturu založenou na komponentách:
    - Architektura přináší velké výhody. Díky ní je možno projekt bez problémů rozdělit na menší části a přidělit více vývojářským týmům, čímž umožňuje postupnou tvorbu systému a zapracování změn je o mnoho jednodušší,
- virtuální modelování:
    - V metodice RUP se využívá modelovacího jazyku UML, které zpřehledňuje jednotlivé etapy vývoje, dále umožňuje udržet konzistenci mezi modelem a vlastní implementací,
- kontrola softwarové kvality:
    - Kontrolou softwarové kvality se má na mysli především kontrola funkcionality, spolehlivosti a výkonu,
- řízení změn:
    - Při potřebě změny je možné ji provést v rámci celého vývoje SW, což na druhou stranu přináší velké problémy, co se týká dokumentace.

Schéma projektu podle metodiky RUP nám znázorňuje následující obrázek, kde na horizontální ose je vyznačen čas a na vertikální jsou naneseny jednotlivé disciplíny, které na sebe navazují v rámci životního cyklu vývoje IS. Celý vývoj je rozdělen na čtyři fáze, které jsou prováděny v několika dílčích krocích (tzv. iteracích) a které mají jasně definované cíle.[73]

[73] Aldorf, F. Vývoj aplikací s použitím metodiky Rational Unified Process. Diplomová práce, VŠE

Obr. 18: Schéma projektu podle metodiky RUP



Nyní si rozebereme podrobněji jednotlivé fáze.

Cílem fáze zahájení je definovat cíle projektu, jeho požadavky, sestavit harmonogram projektu, odhad nákladů (lidských, časových, finančních) a ověřit, zda lze požadavky splnit s danou technologií. Výsledkem fáze je rozhodnutí, zda je možné projekt za daných zdrojů realizovat.

Přípravná fáze definuje architekturu systému, tedy vytvoření prototypu, ověření architektonických principů a zpřesnění plánu realizace systému a definování komponent pro znovupoužití.

Úkolem konstrukční fáze je návrh systému, jeho realizace zahrnující testování.

Fáze předávání zajistí možnost využívat systém, tzn. její součástí je také školení uživatelů, help desk a předání dokumentace[74].

8.5.4. Metodika EUP

Metodika EUP[75] je rozšířením metodiky RUP, zahrnuje navíc dvě nové fáze (používání a vyřazení) a obsahuje několik nových disciplín (např. portfolio management, opětovné použití, HR management, podnikovou architekturu a administraci, proces zlepšování SW, atd.).

---

[74] Buchalcevová, A. Návrh metodického rámce IS/ICT: Metodiky budování IS/ICT. KIT VŠE 2004. str. 19

[75] Ambler, S. W. Enterprise Unified Process (EUP) [online]. Agile Strategies for Enterprise IT. Ambysoft Inc. 2007 Dostupné na: http://www.enterpriseunifiedprocess.com/

8.5.5. OPEN: Object-oriented Process, Environment and Notation

Object-oriented Process, Environment and Notation je veřejně přístupnou metodikou, která podporuje kompletně celý životní cyklus vývoje IS. Metodika je velmi podrobná, ale je zaměřená pouze na úroveň projektu a orientovaná jen na orientovaný a komponentový vývoj nového řešení[76].

8.5.6. PDIT

Metodika PDIT[77] byla vyvinuta Katedrou informačních technologií Vysoké školy ekonomické spolu s dnes již neexistující firmou ITC PragoDATA pro vývoj aplikací ve specifických podmínkách.

[76] Buchalcevová, A. Návrh metodického rámce IS/ICT: Metodiky budování IS/ICT. KIT VŠE 2004. str. 44 - 45

[77] Řepa, V. Analýza a návrh informačních systémů. Ekopress, Praha 1999, ISBN: 80-86119-13-0. str. 249

8.5.7. Agilní metodiky

Jak již bylo řečeno na začátku, softwarová krize byla důvodem vzniku mnoha nových pohledů a změn v softwarovém inženýrství a jednou z nich jsou také agilní metodiky. Většina projektů končila se zpožděním, s překročeným rozpočtem či s úplně jinou funkcionalitou než zákazník očekával. Začaly vznikat agilní metodiky a přesto, že každá metodika byla trošku odlišná, vzhledem k odlišnosti projektů, přesto byly v základu stejné.

Název agilní metodiky pochází z anglického adjektivum agile, které v českém překladu znamená hbitý, čilý, svižný či bystrý. Vzhledem k tomu, že si nedokážu představit, že bychom se zabývali hbitými či bystrými metodikami, lze dost dobře pochopit, proč se ujal právě výraz agilní.

Agilní metodiky představují v podstatě reengineering procesů pro budování IS/ICT, zaměřují se na vývoj nového řešení, nikoli na údržbu a provoz.

Tyto metodiky pocházejí z poznání, že jediný způsob, jakým lze ověřit správnost navrženého systému, je co nejrychleji jej vyvinout, předložit zákazníkovi a na základě zpětné vazby jej upravovat.[78]

V roce 2001 byl podepsán tzv. Manifest agilního vývoje softwaru skupinou sedmnácti programátorů.[79] Manifest hovoří o objevování lepších způsobů vývoje softwaru tím, že jej vytváříme, čímž pomáháme i ostatním. Během vývoje se učíme preferovat[80]:

- lidi a jejich vztahy před procesy a nástroji,
- plně funkční software před vyčerpávající dokumentací,
- spolupráci se zákazníkem před jednáním o smlouvě,
- zohlednit změny před dodržením plánu.

Mezi rysy manifestu tedy patří[81]:

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

---

[78] Kadlec, V. Agilní programování: Metodiky efektivního vývoje softwaru. Computer Press, Brno 2004, 1. vyd. ISBN: 80-251-0342-0. str. 118

[79] Beck, K. Beedle, M. Bennekum, A. Cocburn, A. Cunningham, W. Fowler, M. Greening, J. Highsmith, J. Hunt, A. Jeffries, R. Kern, J. Marick, B. Martin, R. C. Mellor, S. Schwaber, K. Sutherland, J. Thomas, D. Manifesto for Agile Software Development [online]. 2001., Dostupné na: http://agilemanifesto.org/

[80] Vlk, T. Manifest agilního vývoje softwaru z osobní perspektivy [online]. Tril 2008. Dostupné na http://www.tril.cz/manif.html

[81] Co jsou agilní metodiky vývoje software [online]. Reengine 2008. Dostupné na: http://www.reengine.cz/index/agilni_metodiky.do

- důraz na výslednou hodnotu pro zákazníka před dokumenty a papírováním.

Všechny agilní metodiky odpovídají manifestu a je pouze na každém z nás, jakou z nich si vybereme. V následujícím textu se zaměříme na nejznámější agilní metodiky.

Buchalcevová se zmiňuje o osmi nejčastěji používaných metodikách v České republice, kterými jsou:[82]

- Dynamic Systems Development Method (DSDM),
- Adaptive Software Development (ASD),
- Feature–Driven Development (FDD),
- Extrémní programování (Extreme Programming, XP),
- Lean Development,
- Scrum ,
- Crystal metodiky,
- Agilní modelování (Agile Modeling).

My se v následujícím textu budeme věnovat těm nejpoužívanějším. Začneme těmi nejpopulárnějšími, kterými jsou Extrémní programování a Scrum.

8.5.7.1. XP: Extrémní programování

Metodika XP známá pod názvem Extrémní programování je j Jednou z nejznámějších agilních metodik je extrémní programování, které je vyvedeno skutečně až do extrému, neboť např. jedinou psanou dokumentací je programovací kód sám o sobě. Na druhou stranu se klade obrovský důraz na jeho vysokou čitelnost a jednoduchost. Zároveň dochází k téměř denním změnám požadavků funkčnosti aplikace ze strany zákazníka.

Extrémní programování (Extreme Programming), známé pod zkratkou XP, je metodikou Kenta Becka, která obsahuje best practices vývoje IS v podobě metod.[83]

Pilíři XP je především[84,85]:

- Párové programování.
    - Párové programování znamená, že dva společně pracující programátoři se vzájemně kontrolují, čímž dochází k minimalizaci chyb. Další výhodou je, že pokud se z nějakého důvodu jeden z programátorů nemůže dostavit, druhý kód zná.

---

[82] Buchalcevová, A. Stav používání agilních metodik v ČR. Systémová integrace 2006, cis. 4 ,ISSN 1801-1578

[83] Copeland, L. Extreme Programming, Computerworld 2001, ISSN 0010-4841

[84] Ringoš, J. Extrémní programování. PC Svět, srpen 2004 , ISSN 1213-6042

[85] Hayes, S. Andrews, M. An Introduction to Agile Methods [online]. Wry Tradesman 2004. Dostupné na: http://www.wrytradesman.com/articles/IntroToAgileMethods.pdf

- Neustálé testování.
    - Může nastat případ, kdy starý kód byl v minulosti otestován, ale nově přidanou funkčností již nepracuje, jak by měl. Z toho důvodu se provádí stále dokola testování i starého kódu při každé podstatné změně. Navíc testování provádí zákazník sám osobně.
- Neustálé přepracovávání designu.
    - S každou významnou změnou je třeba také změnit i design, protože starý již neodpovídá nově vzniklé situaci.
- Jednoduchost programů.
    - Říká se, že v jednoduchosti je elegance, což platí i v případě XP. Kódy píšeme čitelně s maximální jednoduchostí pro současnou funkcionalitu. Zakomponování změn je pak velmi jednoduché a rychlé, protože se vyhneme zbytečnému přepracování částí, které sice neměly žádný význam, ale již v programu existovaly.
- Samostatnost a integrace.
    - Doporučení zní co nejvíce oddělovat veškeré související části od sebe, tzn. rozdělovat programový kód do knihoven či komponent, tedy do samostatně použitelných souborů. Důležité pravidlo pro integraci: zakomponování starého kódu do nového již funkčního je možné jen a pouze za podmínek, že je nový kód odladěn, otestován na všechny potencionální chyby a plně funkční.
- Krátké iterace projektů.
    - Krátkými iteracemi projektů se má na mysli čas mezi dvěma vývojovými verzemi kompletního projektu. Tyto verze musí být otestovány, což v rámci XP trvá dny, maximálně týdny a v rámci obvyklých projektů týdny, měsíce a roky.
- Příběhy.
    - Příběh popisuje stručný popis toho, co je třeba v projektu přidat, odebrat či změnit. Příběhy by se měly archivovat a v žádném případě je nemazat, v nejlepším případě v rámci databáze a třídit je podle priority. Příběhy slouží pouze pro vnitřní užití, nikoliv pro zákazníka.
- 40-ti hodinový doporučený limit.
    - Limit je nutným omezení pro týdenní práci programátora, který docílí spokojenosti a odpočinku pro další práci. Tento limit lze upravit dle potenciálních schopností a výdrži. Občas je zapotřebí pracovat přesčas, ale je třeba dbát na to, abychom tak nepracovali několik týdnů za sebou a navíc, ne vždy lze daný problém vyřešit neustálými hodinami přesčas, tedy pokud se zvýší četnost hodin přesčas, je to znamením, že problém musíme řešit jinak.
- Standardy.
    - Jsou důležité pro psaní kódu a při jejich dodržování by nemělo být zjistitelné, který programátor daný kód psal. Nejdůležitějším standardem XP je vyhnout se duplicitám kódu.

Základními principy extrémního programování je okamžitá reakce na nově vzniklé požadavky v rámci dní, požadovaná jednoduchost, postupné změny, kvalitní práce, učení se samostatně rozhodnout o stylu programování, nízké počáteční investice, otevřená upřímná komunikace, přijímání odpovědností a kolektivní vlastnictví, realistické odhady psaní kódů, přizpůsobení pravidel extrémního programování místním podmínkám, atd.

XP je velmi dobře propracovaná metodika[86], jejím jádrem je dvanáct praktik, které se vzájemně doplňují a podporují[87]. Základem vývoje je příběh (tzv. User Story). Výhodou metodiky je velké množství zdrojů jak na českém tak na zahraničním internetu. Nevýhodou metodiky jsou velmi striktně definovaná pravidla. Metodiku je třeba implementovat kompletně celou, což nebývá vždy možné. Extrémní terminologie metodiky může vyvolat u manažerů představu "*divokého, nekontrolovaného procesu či chaosu s nejasnými zárukami.*" Metodika je v České republice i v zahraničí velmi používána. Je vhodná pro týmy, který není třeba řídit moc pevně, typickou vlastností je kolektivní vlastnictví týmu.[88]

Extrémní programování je velice obsahově širokou a velmi zajímavou záležitostí. Má práce se ho jen tak zlehka dotkla z důvodu vytvoření přehledu agilních metodik. Jen bych závěrem připomněla, že stejně jako jakákoliv jiná metodika, ani extrémní programování není vhodné pro všechny typy vývoje IS.

8.5.7.2. Scrum

"Název metodiky vznikl z anglického slova Scrum, jehož význam spadá do ragby a označuje skrumáž, mlýn, projevující se kumulací několika (typicky mnoha) hráčů na jednom místě za účelem společného dotlačení míče na požadovanou pozici. Paraela s vývoj softwaru je jednoznačná: cílem vývojového týmu není také nic jiného než "dotlačení míče" na požadovanou pozici, tedy dokončení aplikace v podobě, jakou vyžaduje zákazník. Celý vývojový proces si v metafoře metodiky SCRUM lze představit jako ragbyový zápas a vítězstvím není nic jiného nežli spokojený zákazník."[89]

Termín SCRUM pochází již z roku 1986 ze studie Nonaka a Takeuchiho[90], která byla zveřejněna v Harvard Business Review. Tato studie pojednává o tom, že pokud IS vyvíjí týmy z různých oddělení organizace (tzv.cross-functional teams), můžeme dosáhnout nejlepších výsledků v historii.

V roce 1993 vyvinul Jeff Sutherland proces Scrum ve společnosti Easel Corporation a použil studii Nonaka a Takeuchiho jako základ pro vytvoření týmů

---

[86] Beck, K. Extrémní programování. Grada, Praha 2002, ISBN. ISBN 80-247-0300-9

[87] Pergl, R. Analýza vnitřních vazeb principů metody extrémního programování. Sborník

[88] Pergl, R. Struska, Z. Čím mohou přispět nejznámější agilní metodiky ke zlepšení vývojového procesu. ČZU PEF Praha 2002. str. 799

[89] Kadlec, V. Agilní programování: Metodiky efektivního vývoje softwaru. Computer Press, Brno 2004, 1. vyd. ISBN: 80-251-0342-0. str. 147

[90] Takeuchi, H. and I. Nonaka. The New New Product Development Game. Harvard Business Review 1986

a převzal také jejich pojmenování Scrum jako označení tohoto procesu jako celku.

Ken Schwaber jako první v roce 1995 formalizoval metodiku Scrum v knize Scrum at OOPSLA 1995 jako proces pro softwarový průmysl na celém světě.[91]

Základem vývoje v metodice SCRUM je tzv. Sprint neboli běh. Scrum je jednoduchým rámcem, který funguje na principu "zkontroluj a přizpůsob", má tři typy rolí, tři typy funkcí a tři typy artefaktů navržených tak, aby přinášely fungující software v jednotlivých 30ti denních bězích (Sprintech).

Cílem metodiky je takové organizování týmů, které vede k vyšší produktivitě a kvalitě. Umožňuje týmům, aby si sami zvolily množství práce, kterou by chtěli vykonat a také způsob, jak ji provést co nejlépe. Tím se vytváří velmi příjemné a motivující produktivní pracovní prostředí. Scrum je založen na podnikových hodnotách, zlepšování užitečnosti vyvíjených aplikací a zvyšování příjmů. Je navržen takovým způsobem, aby odpovídal neustále se měnícím požadavkům v rámci průběhu procesu vývoje, umožňuje upřednostnit požadavky zákazníků v reálném čase, čímž jasně a zřetelně dokáže stanovit, co zákazník opravdu požaduje v době dodání a minimalizují se tak tak náklady spojené s odstraněním chyb, které se řeší okamžitě.[92]

V této metodice má každý vývojář jasně vymezen svůj kód, tedy je na první pohled zřejmé, kdo má odpovědnost za daný objekt odpovědnost, čímž je umožněna specifikace, zřejmou nevýhodou je problém s předáním následníkovi v případě odchodu zaměstnance. Výhodou metodiky je specializace na řízení týmu, řízení rizik a je vhodnější pro firmy, které potřebují "pevnější vedení" než při XP.

8.5.7.3. DSDM: Dynamic Systems Development Method

Metodika DSDM vznikla ve Velké Británii v první polovině 90. let.I vzhledem k dlouhověkosti metodiky, je metodika neustále přizpůsobována nejnovějším trendům, znalostem a zkušenostem. Její rozvoj a rozšiřování DSDM konsorcium. Metodiku lze získat za členský poplatek u konsorcia (akademický, vládní, firemní,...) a jeho cena se pohybuje od desítek po tisíce liber.[93]

Metodika má velmi propracovaný iterativní vývojový cyklus vně i uvnitř fází, umožňuje se vracet mezi jednotlivými fázemi. Ve srovnání s ostatními agilními metodikami má nejpropracovanější školení a dokumentaci a je oblíbená v Evropě i v USA. Metodika je zaměřena na softwarově inženýrskou oblast, především na vývoj nového řešení, kombinuje přístup rychlého vývoje aplikací s objektově orientovaným vývojem. Základní technikou používanou při

---

[91] Schwaber, K. Sutherland, J. Scrum Development Process. OOPSLA Business Object Design and Implementation Workshop 1997. Springer: London

[92] Scrum Concept [online]. ScrumAlliance. Dostupné na: http://www.scrumalliance.org/view/scrum_concept

[93] Pergl, R. Struska, Z. Čím mohou přispět nejznámější agilní metodiky ke zlepšení vývojového procesu. ČZU PEF Praha 2002. str. 801

analýze a návrhu je prototypování. Přínosem metodiky je řízení jejího rozvoje, propagace, školení a implementace.[94]

8.5.7.4. Metodiky Crystal

Autorem metodik Crystal je Alistair Cockburn, jedná se o rodinu metodik, která je určena pro různé typy projektů. Každá metodika je určena pro projekt určité důležitosti a rozsahu. Metodiky jsou poté dále zaměřené na maximalizaci produktivity, sledovatelnosti a jiných parametrů. Výhodou této metodiky je tedy především její konfigurovatelnost, ze všech agilních metodik je nejlépe škálovatelná, neboť rozlišuje projekty podle počtu zúčastněních. Metodika preferuje osobní pravidelné schůzky před explicitním řízením rizik. Metodika je vhodná pro méně zkušené, kteří potřebují pomoc při nastavování procesů, pravidel, dokumentů apod.[95]

8.5.7.5. Feature Driven Development

Metodika Feature Driven Development je považována za konzervativnější metodiku a v porovnání s ostatními agilními metodikami má nejblíže ke klasickému přístupu. Tato metodika definuje své vlastní pojmy, procesy a klade velký důraz na modelování. Základem metodiky jsou vlastnosti systému (tzv. Features), což umožňuje provádění průběžných kontrol a kvantifikaci vývoje.[96]

Metodika zavádí vlastnictví tříd a jejich odpovědnost. Tím se vytváří flexibilně, což umožňuje zvládání i robustnějších projektů, u kterých může u ostatních agilních metodik nastat problém.

Nevýhodou metodiky je, že nepočítá s možností změny v harmonogramu jako např. metodika SCRUM, což může vést k fatálním problémům.[97]

8.5.7.6. Lean Development

Metodika Lean Development vznikla po válce v Japonsku, kdy bylo zapotřebí zefektivnit výrobu automobilů. I když se liší automobilový průmysl od softwarového inženýrství, přesto se našlo mnoho analogií a inspirací, důkazem čehož je vznik této metodiky.

Čím je tato metodika známá, je její "strategičnost" na rozdíl od ostatních metodik. Metodika se zabývá především pojmy "hodnota a plýtvání", snaží se o zefektivnění procesů odstraněním zbytečných činností a zaměřuje se na efektivní subdodávky a používání komponent. Metodika poskytuje velmi příjemné a motivující pracovní prostředí i když je v některých ohledem

---

[94] Buchalcevová, A. Návrh metodického rámce IS/ICT: Metodiky budování IS/ICT. KIT VŠE 2004

[95] Pergl, R. Struska, Z. Čím mohou přispět nejznámější agilní metodiky ke zlepšení vývojového procesu. ČZU PEF Praha 2002. str. 800 - 801

[96] Felsing J, Palmer, M. S. R. a Practical Guide to Feature - Driven Development. Prentice Hall 2002. ISBN 0130676152

[97] Buchalcevová, A. Metodika feature-driven development neopouští modelování a procesy, a přesto přináší výhody agilního vývoje. sborník konference Tvorba softwaru. Ostrava 2005

ve srovnání s ostatními metodikami opatrnější, např. co se týče udělování pravomocí zaměstnancům.[98]

8.6. Závěr

V této kapitole jsme si ukázaly, že metodiky nám pomáhají při vývoji IS díky radám, jak postupovat v jednotlivých fázích životního cyklu, např. jaké používat postupy, pravidla, jaké dokumenty, jakých nástrojů, technik a metod.

Metodiky nám pomáhají odhalit chyby již v počáteční fázi jejich vzniku, dokonce dříve, než je máme sami možnost zjistit, což přináší obrovské úspory na jejich odstraňování.

Existuje velké množství kritérií, podle kterých lze metodiky členit, my jsme se zaměřily na členění z hlediska zaměření na metodiky obecné a specifické, dále z hlediska váhy na tradiční a agilní.

Obecné metodiky chápou proces vývoje velmi obecně a je třeba je přizpůsobit podle podmínek dané firmě a projektu. Příkladem je metodika RUP a EUP. Naopak, speciální metodiky jsou přizpůsobeny na míru přímo určité firmě a jsou určeny pro určitý typ projektu pro určitou situaci a další přesně definované podmínky.

Tradiční metodiky jsou jak historickými „tradičními" přístupy, které se používaly v minulosti, ale také se používají i v současně době. Jejich příkladem je klasická a Yourdonova metoda, Code and Fix model, metoda tunel, byrokratická metoda, Stagewise model, metoda vodopád (fontána), přírůstková metoda, spirálový model a model Win-Win. Tradiční metodikou je model zralosti (SW-CMM model), metodika RUP a EUP, OPEN a PDIT.

Agilní metodiky představují v podstatě reengineering procesů pro budování IS/ICT, zaměřují se na vývoj nového řešení, nikoli na údržbu a provoz. Tyto metodiky pocházejí z poznání, že jediný způsob, jakým lze ověřit správnost navrženého systému, je co nejrychleji jej vyvinout, předložit zákazníkovi a na základě zpětné vazby jej upravovat. Nejznámějšími příklady těchto metodik je Extrémní programování (XP), metodika SCRUM, Dynamic Systems Development Method (DSDM), metodiky Crystal, Feature Driven Development a Lean Development.

Tradiční metodiky vycházejí z požadavku, že funkcionalita je fixní a již se nemění, čas a zdroje jsou variabilní konstantou. U agilních metodik je funkcionalita proměnlivá, ale čas a zdroje jsou považovány za fixní.

[98] Poppendieck, M. Poppendieck, T. Lean Software Development: An Agile Toolkit for Software Development Managers. Addison-Wesley 2003. ISBN 0321150783

9. CASE nástroje

9.1. Úvod

V této části si vysvětlíme pojem CASE nástroje, podíváme se blíže na jejich členění, řekneme si něco blíže o důležité součásti CASE nástrojů – repository a o jejích komponentách.

9.2. Pojem CASE nástroje

Zkratkou CASE se v odborné literatuře označuje pojem Computer Aided Software Engineering[99],[100], ale dnes již bývá velmi často používán pojem Computer Aided System Engineering[101],[102]. Původní označení se postupně upravilo ze Software na System, neboť dnešní CASE nástroje řídí nejenom vývoj softwaru jako takového, ale také již obsáhlejší systémy IS/ICT. V současnosti můžeme v české i zahraniční literatuře nalézt obojí označení a tudíž oba pojmy můžeme považovat za synonyma.

CASE můžeme volně přeložit jako počítačem podporované softwarové inženýrství pro vývoj příp. údržbu počítačových systémů. Tyto nástroje využívají metodiky pro organizování, řízení a vývoj IS a to především rozsáhlých složitých projektů, které zahrnují mnoho softwarových komponent, na kterých spolupracuje velké množství lidí. Umožňují návrhářům, programátorům, testerům, projektantům, manažerům, a dalším rolím v rámci vývoje, sdílet společný přehled o projektu v každé fáze jeho vývoje[103].

---

[99] Řepa, V. Chlapek, D. Materiály ke strukturované analýze. Vysoká škola ekonomická, Praha 1997, ISBN 80-7079-260-4. str.125

[100] Houser, P. Computer Aided Software Engineering (CASE). Business World, únor 2007, ISSN: 1682-3257

[101] Bitpipe: The TechTarget Library of White Papers, Product LIterature, Webcasts and Case Studies [online]. Dostupné na: http://www.bitpipe.com/tlist/Computer-Aided-Software-Engineering.html (navštíveno 15. 8. 2008)

[102] Carnegie Mellon: Software Engineering Institute. What is a CASE Environent [online]. Carnegie Mellon University 2007. Dostupné na: http://www.sei.cmu.edu/legacy/case/case_whatis.htm

[103] Meecham, B. CASE [online]. Midmarket CIO Definitions 2005. Dostupné na: http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci213838,00.html#

Obr. 19: Přínosy softwarových best practices[104]

CASE nástroje zefektivňují používání metodik a automatizovaných nástrojů, čímž vytváří synergický efekt, neboť spojují best practices, ověřené postupy, (předpokládám) spolu s celkovým pohledem na vybranou dimenzi (část fáze) či na všechny dimenze životní fáze vývoje IS v rámci metodik.

9.3. Proč potřebujeme CASE nástroje?

Rubin popisuje, co způsobuje nejčastěji kolaps při vývoji softwaru. Došel k závěru, že každá etapa vývoje v sobě skrývá potenciální možnost selhání[105]:

- 16% analýza,
- 17% design,
- 34% testování programu a jednotkový test,
- 18% systémové a integrační testy,
- 8% dokumentace,
- 7% implementace, instalace.

Z výše uvedeného je zřejmé, že celý proces vývoje IS je rizikový a tudíž je třeba ho řídit komplexně.

---

[104] Panter, S. Software Best Practice (ESSI) [online]. ESSI 1997. Dostupné na: http://cordis.europa.eu/esprit/src/essi.htm

[105] Rubin, H. Worldwide Benchmark Project Report. Rubin Systems Inc. 1995

Ve svých počátcích sloužily CASE nástroje jako okrajová pomůcka pro softwarové inženýry pro vývoj aplikací. Tehdy tyto nástroje podporovaly dílčí činnosti jako např. kompilaci zdrojového kódu nebo kontrolu syntaxe.

Postupem času byly vyvinuty metodiky pro vývoj IS a bylo zapotřebí více sofistikovaných nástrojů, které by pokryly požadavky těchto nových metodik (zejména možnost grafického zobrazení diagramy, anotace, potřeba vedení a správy projektové dokumentace). Takto se CASE nástroje dostaly na takovou úroveň, že byly schopny podporovat celý proces vývoje aplikace od analýzy až po implementaci a testování. Uvádí se[106], že již na počátku 80. let 20. století došlo v CASE nástrojích „*k dosažení kritické kvality v metodách, organizaci práce a technologiích potřebných při vývoji IS.*"

Původně sloužily CASE nástroje pouze k řízení vývoje informačního softwaru, proto se také používala zkratka Computer Aided Software Engineering. Jak se nástroje postupem času rozvíjely, zahrnovaly navíc kromě vývoje systému i plánování, provoz, údržbu a rozvoj systému a začaly podporovat jak strategické rozhodování na počátku projektu, tak operativní činnosti, které souvisejí s provozem systému a řízením jeho změn a rozvoje. V té době se začal užívat název Computer Aided System Engineering.

Po určité době došlo z důvodu nárůstu objemu funkcionality nástrojů k oddělení modelovací a řídící vět, z nichž vznikly samostatné skupiny produktu.

Dnešní CASE nástroje umožňují následující činnosti [107],[108]:

- tvorba konceptuálního i fyzického datového modelu,
- objektové modelování,
- kontrolní mechanismy,
- porovnávání modelů,
- generování výsledného kódu,
- podpora spolupráce při vývoji,
- automatické vytváření dokumentace,
- reengineering,
- podpora návrhu vícerozměrných schémat,
- procesní modelování,
- vytváření datových slovníků,
- podpora životního cyklu,

---

[106] Řepa, V. Analýza a návrh informačních systémů. Ekopress, Praha 1999, ISBN: 80-86119-13-0. str. 16

[107] Kadlec, V. Case a zbytek světa II., požadavky [online]. Databázový svět 2002. Dostupné na: http://www.dbsvet.cz/view.php?cisloclanku=2002062801

[108] Kadlec, V. Case a zbytek světa II., požadavky [online]. Databázový svět 2002. Dostupné na: http://www.dbsvet.cz/view.php?cisloclanku=2002070104

- správa požadavků uživatelů.

CASE nástroje obsahují dvě základní myšlenky[109]:

1) počítačová podpora při vývoji softwaru a/nebo jeho údržba,

2) inženýrský přístup k vývoji a udržování softwaru.

9.4. Typy CASE nástrojů

V této podkapitole se podíváme blíži na základní členění CASE nástrojů podle jednotlivých fází životního cyklu. Díky tomuto členění si dokážeme lépe uvědomit, které činnosti mají být podporovány daným nástrojem.

9.4.1. Vertikálně a horizontálně orientované nástroje

Existuje rozličné spektrum CASE nástrojů. Některé CASE nástroje pokrývají všechny fáze životního cyklu (tzv. vertikálně orientované nástroje[110]), jiné pokryjí jen jeho část, tedy jen jednu či několik specifických fázi životního cyklu (tzv. horizontálně orientované CASE nástroje). Běžnější je ale, že CASE nástroje pokrývají jen určité specifické činnosti, což vyplývá z toho, že každá fáze zahrnuje jiné činnosti a proto je pro každou fázi většinou použita i jiná metodika a tím pádem i jiný nástroj, který ji podporuje.

9.4.2. Nástroje z hlediska podpory

Podle toho, zda CASE nástroj podporuje jen určitou činnost či celou fázi nebo celý cyklus, rozlišujeme:

- CASE nástroje určené pro jeden konkrétní účel.
    - Podporují jen určitou dílčí činnost, která je součástí určité fáze.
    - Příkladem mohou být nástroje určené specielně pro:
        - projektování reality (Business Systém Planning Tools),
        - řízení projektů (Project Management Tools),
        - podpůrné činnosti (Support Tools),
        - analýzu a návrh (Analysis & Design Tools),
        - programování (Programming Tools),
        - integraci a testování (Integration & Test Tools),

---

[109] Wiley, J. Putting the Software Engineering into CASE by K. Robinson, published by John Wiley and Sons Inc. New York 1992

[110] Jánský, V. Kříž, P. Šebelík, J. Podpora plánování a řízení projektů v CASE nástrojích [online]. prosinec 2005. Dostupné na: http: //panrepa.org/CASE/CASE_vs_RIP.pdf. str. 3

        - prototyping (Prototyping Tools),
        - údržbu (Maintenance Tools).
- CASE nástroje určené pro jednu konkrétní fázi.
    - Nástroj podporuje několik fází životního cyklu. Obvykle je každá fáze podporována jinou metodikou a tedy i jiným v tomto případě „podnástrojem“. Tyto „podnástroje“ jsou mezi sebou vzájemně integrovány a skládá se z nich daný CASE nástroj.
    - Metodika znamená pro firmu nemalé náklady, proto se snaží jejich počet minimalizovat. Tedy i CASE nástroje se snaží být nezávislé na konkrétní metodice, jako příklad můžeme uvést Power Designer.
- I-CASE – plně integrované nástroje
    - I-CASE[111] neboli integrovaný CASE je takový CASE nástroj, který pokrývá celý životní cyklus vývoje IS. Jedná se o sadu produktů, které jsou navrženy tak, aby umožňovaly vzájemnou komunikaci v rámci jednotlivých nástrojů a přesnost dat z analýzy, designu a vývoje.[112]

CASE nástroje byly původně rozlišovány[113] podle fází životního cyklu na:

- Upper CASE
    - Jsou využívány k podpoře činností, které se zabývají vytvořením informační strategie organizace a úvodní studie informačního systému.
    - Tyto nástroje můžeme zařadit do fáze Informační strategie a úvodní studie v rámci vývoje IS.
- Middle CASE
    - Exaktní metodiky a techniky, které podporují analytickou a logickou část návrhu systému.
    - Tyto nástroje můžeme zařadit do fáze Globální a detailní návrh.
- Lower CASE
    - Podporují implementaci z výsledků analýzy a z logického návrhu systému. Bývají závislé na konkrétním prostředí.
    - Tyto nástroje můžeme zařadit do úrovně Implementace).

---

[111] Šenovský, P. Case systémy [online]. VŠB-TUO FBI 2006. Dostupné na: http://homel.vsb.cz/~sen76/case/2CASE.doc (navštíveno 2. 7. 2007)

[112] Modell, M. E. a Professional's Guide to Systems Analysis: Glossary of Terms and Concepts [online]. McGraw-HillBook Company New York 2007. Dostupné na: http://www.martymodell.com/pgsa2/pgsa-glossary.html (navštíveno dne 2.7.2007)

[113] Řepa, V. Chlapek, D. Materiály ke strukturované analýze. Vysoká škola ekonomická, Praha 1997, ISBN 80-7079-260-4. str.126 - 127

Časem bylo členění ještě upřesněno a rozšířeno o úroveň Pre CASE a Post CASE. Toto členění bylo ustáleno a používá se do dneška.

CASE nástroje členíme na[114] [115]:

- Pre CASE
    - Jsou využívány k podpoře vytvoření business plánu.
    - Cílem je vytvoření globální strategie.
    - Tyto nástroje můžeme zařadit do úrovně Globální strategie v rámci vývoje IS.
- Upper CASE
    - Podporují plánování, specifikaci požadavků, modelování organizace podniku a globální analýzu informačního systému.
    - Použitými nástroji mohou být Data Flow Diagram (DFD) a Entity Relationship Diagram (ERD), obojí bez podrobných atributů, nástroje pro řízení projektů, nástroje pro sledování ekonomických ukazatelů. Nezbytný je také popis základních vlastností systému pomocí objektově orientovaného modelování.
    - Cílem je pochopit určitou oblast a specifikovat systém jako celek.
- Middle CASE
    - Slouží k podpoře vytvoření podrobné specifikace požadavků, vytvoření vlastního návrhu systému, dokumentace systému a pro vizualizaci systému.
    - Uvádí se, že se jedná o nejúspěšnější třídu CASE nástrojů.
    - Použitými nástroji mohou být Data Flow Diagram (DFD) s podrobným popisem procesů, datových úložišť, dále Relationship Diagram (ERD), obojí již s podrobnými atributy. Dále obsahuje diagramy tříd, instancí, přechodové diagramy atd. pro objektově-orientovanou analýzu a návrh (OOAN).
    - Cílem je zformalizovat specifikace a návrh s možností snadných změn a vytvoření modelu pro generování návrhu.
- Lower CASE
    - Jsou nástroje, které podporují kódování, testování a údržbu, slouží tedy k urychlení vývoje aplikace a k návrhu databáze, čímž zlepšují kvalitu a efektivnost práce nejen programátorů ale i dalších pracovníků. Také slouží jako podpora pro sestavení modelů a jejich dokumentaci, sledují a vyhodnocují metriky, pomáhají plánovat a zjišťovat kvalitu softwaru (sbírají informace

[114] Kaluža, R. Case nástroje [online]. radovan.bloger.cz 2006. Dostupné na: http://radovan.bloger.cz/it/informacni-systemy/case-nastroje

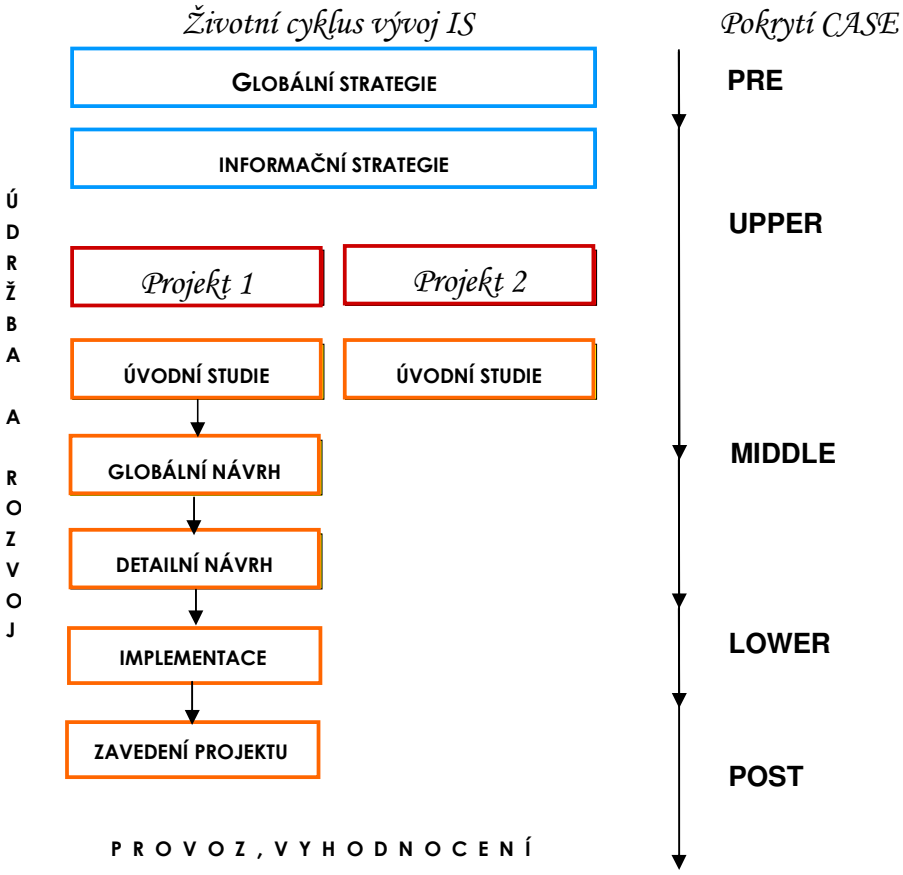
[115] Molhanec, M. Úvod do datového modelování [online]. ČVUT FEL 2008. Dostupné na: http://martin.feld.cvut.cz/~mmm/Vyuka/X13DFA/files/UdDM.pdf

o tom, jak probíhá testování, zjišťují výsledky testování a vyhodnocují je), tedy pro podporu reverzního inženýrství.

- Uvádí se, že tato kategorie CASE nástrojů se často překrývá s funkcemi obecných vývojových prostředí.
- Tyto nástroje jsou určeny pro detailní návrh systému, implementaci, provoz a podporu systému.
- Cílem je sledování a vyhodnocování metrik, plánování, zjištění kvality informačního systému, správa konfigurace a prostředky sledování a vyhodnocování práce informačního systému.

- Post CASE
  - Podporují zavedení do provozu, provozování a údržbu a rozvoj informačního systému.
  - Cílem je podpora organizačních činností.

Obr. 20: Životní cyklus vývoje SW [116]



[116] Molhanec, M. Úvod do datového modelování [online]. ČVUT FEL 2008. Dostupné na: http://martin.feld.cvut.cz/~mmm/Vyuka/X13DFA/files/UdDM.pdf

9.5. CASE a repository

Repository je relační nebo objektová standardní databáze, která plní následující funkce:[117]

- správa složitých dat:
    - udržuje diagramy, textové popisy, naměřené hodnoty,
    - kontroluje integritu dat prostřednictvím kontroly konzistence,
- sdílení informací mezi více subjekty prostřednictvím mechanismů pro selektivní zamykání, řízení změn a kontrolu verzí,
- integrace dat a nástrojů:
    - datový model využívají všechny nástroje,
    - každý nástroj data interpretuje podle svého účelu
- vzájemná integrace dat:
    - zajištění vazeb mezi informačními objekty jako základ pro ostatní funkce manipulace s nimi,
- podpora metodik a procesu:
    - dodržování pravidel návaznosti a pořadí kroků při vývoji modelu,
    - plánování a řízení vývoje,
- standardizace formátu:
    - existuje pouze jeden jediný způsob prezentace dat pro účely komunikace.

Repository také obsahuje data o organizaci, designu aplikace, implementaci, informace o konfiguraci a změnách a o řízení projektu, dále o validaci a verifikaci a také obsahuje dokumentaci.

Dále je nezbytné, abychom při výběru toho správného CASE nástroje zvážili, jaké nástroje bude náš potenciální CASE požadovat, tedy jaké možnosti má nástroj v oblasti repository a zohlednit z toho vyplývající požadavky na:

- nenáročná databázová repository pro malé CASE nástroje,
- náročnější databázová repository pro CASE nástroje střední velikosti,
- velmi náročná databázová repository pro robustní CASE nástroje.

Repository je tedy speciální databáze, do které se ukládají všechny informace týkající se projektu, musí splňovat vývoj náročných a rozsáhlých IS a také zvládat nápor přístupu všech uživatelů. Umožňuje sdílet informace, zajišťovat jejich konzistenci a koordinaci práce, čímž pomáhá k zvýšení efektivity a produktivity při vývoji IS. Je nezbytnou částí CASE nástrojů, ať jde o repository externí či interní (interní je součástí CASE nástrojů a externí není).

---

[117] Dubravec, P. Case nástroje [online]. Dostupné na: http://www.dubravec.cz/dubravcovi/cl000001.htm#a8

Centrální repository umožňuje uchovávat popis všech centrálních komponent systému, které se využívají v rámci celého životního cyklu IS.

9.6. Současnost a budoucnost CASE nástrojů

V současné době můžeme pozorovat všeobecný posun od specializace k integraci. Je to trend, který převažuje v celém odvětví informačních technologií, není tedy překvapivé, že zasáhl také sféru nástrojů pro vývoj IS.

Trendy v oboru CASE nástrojů jsou řízení projektů a unifikace. Projekty v současné době získávají určité funkcionality z oboru řízení projektů a to zejména ty, které se týkají řízení rolí a přidělování zdrojů (času, peněz, pracovníků). Současné metodiky dnes využívají best practices, jedná se o rozsáhlou knihovnu unifikovaných a osvědčených postupů, procesů a doporučení.[118]

Dnešní CASE nástroje se dále rozvíjejí a směřují k tvorbě nových a efektivnějších metodik týkajících se tvorby informačních systémů. Je snaha[119] o to, aby jeden CASE nástroj zachycoval všechny fáze životního cyklu, tedy od strategického plánování a tvorby informační strategie až po zavedení systému do provozu. Trend pokrýt kompletně CASE nástroji celý životní cyklus vývoje IS se nazývá tzv. "Integrated Computer Aided Software Engineering", označované pod zkratkou I-CASE. Současným trendem je představení nových metodik založených na schopnostech I-CASE. Tyto nové metodiky využívají techniky rychlého prototypování k ještě rychlejšímu vývoji aplikací, přičemž splňují požadavky vysoké kvality a nízké ceny. Redukce nákladů se dosahuje díky rychlejšímu vývoji, který umožňuje provádět častěji testování a objevit tak včas chyby, jejichž pozdější odstranění by mohlo být velmi drahé, neboť jak z uvedeného obrázku plyne, čím později jsou chyby objeveny a opraveny, tím je těžší a dražší další vývoj.

Dále je snaha propojit návrh, implementaci a provoz IS/ICT jednotným nástrojem; vytvořit tzv. metasystém, který by dokázal zachytit aktuální stav i plánované stavy IS/ICT. Dalším trendem je zachytit vývoj IS nejen jednou metodikou, ale také prostřednictvím několika metodik. Je to z toho důvodu, neboť společnosti vyvíjejí svůj IS prostřednictvím vstupů a výstupů od různých výrobců a prodejců a je zapotřebí provést systémovou integraci v tomto směru a řídit vývoj prostřednictvím jednotného nástroje.

---

[118] Jánský, V. Kříž, P. Šebelík, J. Podpora plánování a řízení projektů v CASE nástrojích [online]. prosinec 2005. Dostupné na: http: //panrepa.org/CASE/CASE_vs_RIP.pdf

[119] Voříšek, J. Strategické řízení informačního systému a systémová integrace. Management Press, Praha 1999, ISBN: 80-85943-40-9. str. 59

9.7. Přínosy a nedostatky CASE nástrojů

V této podkapitole si uvedeme nejzásadnější přínosy a nedostatky CASE nástrojů.[120]

9.7.1. Přínosy

1) vhodný pro vývoj jakéhokoliv typu software (od standardizovaného po velmi specifický) – kvalitní podpora týmového vývoje
2) snadnější a pohodlnější komunikace při vývoji v rámci týmu či týmů,
3) podpora metod určité konkrétní metodiky,
4) dnešní CASE nástroje podporují veškeré typy modelování (strukturované metodiky, objektově orientované metodiky nebo jejich vzájemnou kombinaci),
5) podpora změny úrovně abstrakce (např. od analýzy přes návrh až ke zdrojovému kódu),
6) celková přizpůsobitelnost (např. možnost přizpůsobit generovanou dokumentaci),
7) automatická generace zdrojových kódů,
8) testování konzistence a validity vytvořeného modelu,
9) synchronizace modelů se zdrojovým kódem,
10) generování dokumentace,
11) zvýšení efektivity,
12) zvýšení kvality,
13) zvýšení produktivity práce,
14) celkové zefektivnění práce,
15) snížení nákladů na vývoj,
16) umožňuje udržení jednotnosti designu během procesů analýzy a návrhu,
17) zjednodušuje provedení změn,
18) umožňuje přepracovat špatně napsaný kód,
19) usnadňuje programovou údržbu,
20) trvá na dodržování softwarových a systémových standardů,
21) podporuje řízení projektu,
22) stará se o centrální úložiště dat a seznam odkazů,
23) zjednodušuje kreslení architektury systémových diagramů,
24) snižuje celkový čas na vývoj aplikací,

[120] McMurtrey, M. E. Teng, J. T. C. Grover, V. Hemant, V. Current utilization of CASE technology. Industrial Management & Data Systems 2000, roc. 100, cisl 1. str. 22 - 30, ISSN 0263-5577

25) umožňuje řešit rozsáhlejší a komplexnější problémy.

26) CASE nástroje dokázaly zkrátit čas nutný pro vývoj IS.

9.7.2. Nedostatky CASE nástrojů

1) nutná školení a vzdělávání osob pracujících s nástrojem,
2) náklady na vývoj a údržbu modelů vytvořených nástrojem,
3) cena za aktualizaci nástroje,
4) nedostatečná integrace s ostatními nástroji,
5) nutnost upravit výsledky nástroje,
6) vygenerovaný kód může obsahovat nadbytečné či naopak nedostatečné informace,
7) mohou vést k závislosti na strukturované metodiky,
8) má limitované funkce,
9) neposkytuje centrální integrované úložiště,
10) vyžaduje znalosti pracovníků ohledně metodik,
11) může způsobit odpor uživatelů,
12) může způsobit změnu rolí a kvality vztahů v rámci pracovního života,
13) velmi těžko lze měřit jejich výhody.

9.8. Co CASE nástroje nejsou

- CASE nástroje nejsou metodologií ani metodikou.
- CASE nástroje nejsou náhradou za vývojová prostředí, i když umí generovat části kódu.
- Používání CASE nástrojů automaticky nezvýší produktivitu a efektivitu vývoje. Je zapotřebí jistá disciplinovanost a určitý přístup, neboť zpočátku je třeba provést mnoho práce, která není nijak vidět.

9.9. Komponenty v CASE prostředí

Komponenty CASE systémů, ze kterých se skládá dané prostředí a tím pádem i daný CASE nástroj, určují funkce a vlastnosti CASE nástrojů, kterými jsou:[121]

- konzistentní grafické ovládací prostředí (podle zásad tvorby GUI) – jednotný vzhled obrazovek, popisků, tlačítek, jednotné ovládání, použití symbolických ikon apod.,
- centrální databáze 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,

[121] Procházka, J. Nástroje CASE [online]. Databázový svět 2004. Dostupné na: http://www.dbsvet.cz/view.php?cisloclanku=2004052702

- textový editor pro popis jednotlivých objektů – pro účely technické a uživatelské dokumentace systému, možnost jejího přímého generování ze systému,
- možnost rychlého návrhu uživatelských obrazovek včetně simulace vstupů a výstupů (je vyžadováno pro prototyping),
- 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 .

Příkladem komponenty je grafické rozhraní, které se skládá ze základních geometrických tvarů (čtverec, kružnice, úsečka, šipka, křivka) s možností uchování. Pokud dojde ke zrušení objektu, tedy k jeho vymazání, konzistence modelu zůstane zachována. Umožňuje také uvést stav do původní, např. při vymazání obnoví původní objekt pomocí funkce undo. Existuje vstupní rozhraní, které umožňuje práce s těmito objekty (klávesnice, myš) a výstupní rozhraní (monitor).

Další důležitou komponentou je slovník CASE systémů. do slovníku se zapisují automaticky např. záznam o objektu diagramu ve chvíli, kdy je vytvořen. Slovník se používá také pro kontroly vazeb mezi entitami pomocí vygenerovaných zpráv, které uživateli oznamují možnost/nemožnost provedení daného kódu. Slovníky podporují generování seznamů datových položek a jejich atributů.

9.10. Závěr

CASE neboli Computer Aided Software Engineering můžeme lze volně přeložit jako počítačem podporované softwarové inženýrství, které slouží pro vývoj příp. údržbu počítačových systémů. Tyto nástroje využívají metodik pro organizování, řízení a vývoj IS a to především rozsáhlých složitých projektů, které zahrnují mnoho softwarových komponent a kterém spolupracuje velké množství lidí. Umožňují návrhářům, programátorům, testerům, projektantům, manažerům, a dalším rolí v rámci vývoje, sdílet společný přehled o projektu v každé fáze jeho vývoje.

Case nástroje se člení na vertikálně a horizontálně orientované. Horizontálně orientované pokrývají jen určitou či několik specifických fází životního cyklu, vertikálně orientované pokrývají všechny fáze životního cyklu. Case nástroje lze dále členit podle životního cyklu na Pre CASE, Upper CASE, Middle CASE, Lower CASE, Post CASE a I-CASE.

Repository je důležitou součástí CASE nástrojů, jedná se o relační či objektovou standardní databázi, která spravuje složitě strukturovaná data, umožňuje sdílení informací mezi více subjekty prostřednictvím mechanismů pro selektivní zamykání, řízení změn a kontrolu verzí, dále integruje data a nástroje, podporuje metodiky a procesy a standardizace formátu. Repository je tedy speciální databáze, do které se ukládají všechny informace týkající se projektu, musí splňovat vývoj náročných a rozsáhlých IS a také zvládat nápor přístupu všech uživatelů. Umožňuje sdílet informace, zajišťovat jejich konzistenci a koordinaci práce, čímž pomáhá k zvýšení efektivity a produktivity při vývoji IS.

Je nezbytnou částí CASE nástrojů, ať jde o repository externí či interní (interní je součástí CASE nástrojů a externí není).

Prostředí CASE nástrojů se skládá z jednotlivých komponent, komponenty tedy určují funkce a vlastnosti CASE nástrojů, kterými jsou především konzistentní grafické ovládací prostředí, centrální databáze, prostředky pro verifikaci konzistentnosti dat a podporu normalizace, textový editor pro popis jednotlivých objektů, generátor zdrojových programů, nástroje pro export a import dat a pro rychlý návrh uživatelských obrazovek včetně simulace vstupů a výstupů.

10. Metamodelování a metainformační systémy

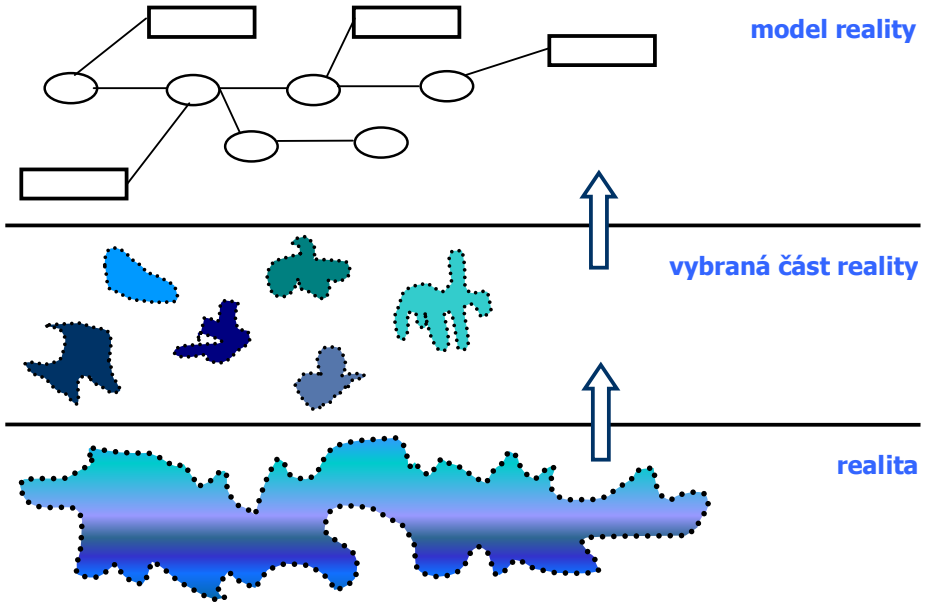
10.1. Úvod

CASE nástroje patří mezi metasystémy, neboť se jedná o modely vývoje samotného informačního systému. Metainformační systémy se používají pro popis průběhu procesů v podniku, pro popis souvisejících dat a popis odpovědností za probíhající procesy a vzniklá data.

Katedra informačních technologií na Vysoké škole ekonomické vyvíjí metainformační systém pod označení MtS, jehož hlavním cílem je „dosáhnout a udržet jednotu IS s cíli a hlavními aktivitami podniky a jednoty všech částí IS/IT navzájem." MtS tedy umožňuje popisovat, analyzovat a řídit IS/IT z pohledu všech významných dimenzí podle metodologie MMDIS z hlediska datového, funkčního a procesního, softwarového, hardwarového, organizačního, personálního, ekonomického, metodického, časového a z hlediska vzájemných vazeb těchto dimenzí.

10.2. Modely a metamodely

Obr. 21: Modelování

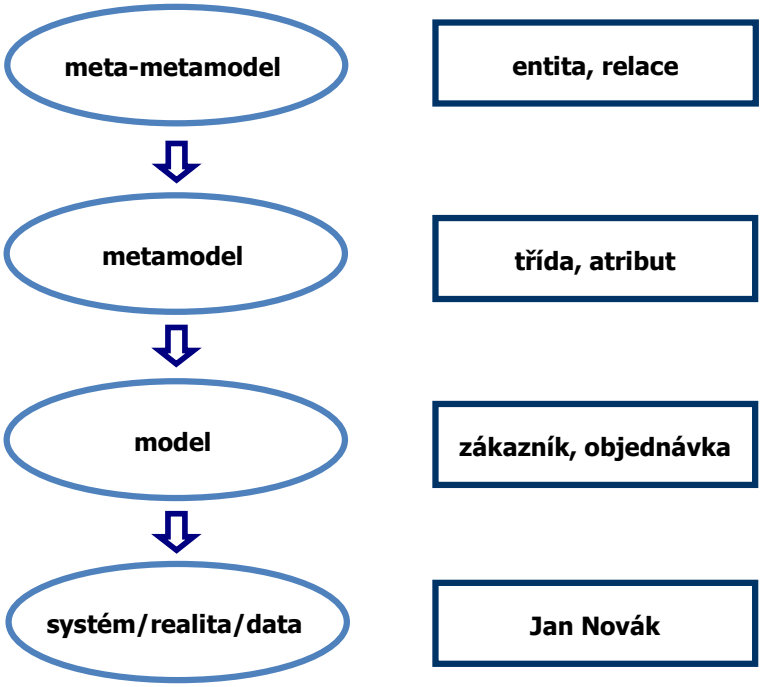


Modelování chápeme jako zjednodušení a abstrahování skutečnosti, metamodelování jako určitou nadstavbu modelování s výraznými prvky abstrakce. Pro lepší představu si uvedeme malý příklad. Modelování přirovnáme procesu tvorby HTML kódu a SGML k metamodelování. SGML nám udává pravidla, které příkazy lze použít při tvorbě webových stránek a aplikací.

Význam předpony "meta" můžeme chápat jako oproštění se od zkoumané problematiky a změnu úhlu pohledu směrem ke globální rovině. To nám umožňuje generalizaci reality, díky čemuž na daný problém díváme shora. V informatice předpona "meta"[122] vyjadřuje zobecnění, kdy metadata jsou data, popisující data. Neboli metamodel je modelem, který popisuje běžné modely. Důvod pro vznik metamodelování byla potřeba popsat odlišné datové struktury, které jsou používány jako komponenty běžně navrhovaných modelů.

Pokud jsme již v minulosti byli schopni zjednodušit realitu do určitého modelu a pro tento model definovat další úroveň abstrakce, pak se můžeme oprávněně domnívat, že platí určitá forma indukční logiky. Z toho vyplývá, že na metamodel můžeme nahlížet jako na strukturu, jejíž komponenty lze zobecnit, čímž vytvoříme jakousi vyšší úroveň náhledu. Tuto úroveň definoval jako meta-metamodel již Bézivin[123] v roce 1998.(viz. obrázek Obr. 22). Pokud vycházíme z logických zásad logické indukce, můžeme kroky pro zobecnění neomezeně opakovat. S každou další úrovní abstrakce ale ztrácíme vypovídací schopnost pro návrh systému a dostáváme se tím spíše do filosofické roviny.

Obr. 22: Model, metamodel, meta-metamodel[124]



[122] Více informací o meta a metamodelování se dočtete v odborné literatuře, např.:

Hřebejk, P. Řepa, V. Příspěvek na konferenci Data-Sem 99. Meta-modelování. praha 1999. Dostupné na: http://nb.vse.cz/~repa/veda/RaD.htm

[123] Bézivin, J. Who is Afraid of Ontologies [online]. Proceedings of OOPSLA´98 Workshop No.25 – CDIF, Vancouver 1998. Dostupné na: http://www.metamodel.com/oopsla98-cdif-workshop/bezivin1/

[124] Bézivin J. Who´s Afraid of Ontologies [online]. Proceedings of OOPSLA´98 Workshop No.25 – CDIF. Vancouver 1998. Dostupné na: http://www.metamodel.com/oopsla98-cdif-workshop/bezivin1/

Metamodel tedy vychází z popisu reality prostřednictvím modelu. Tento model je popsán jako fungující systém, který se využívá při vývoji aplikací, informačních systémů a simulací (např. podnikových systémů). Lze tedy říci, že samotný model je abstrakcí reality, která vytěsňuje nepotřebné a zbytečné informace a spojitosti. Model tak získává vypovídací vlastnosti a zachycuje aktuální stav skutečnosti. Můžeme tak hledat společné vlastnosti u objektů reálného světa, slučovat je dle potřeb či hledat kauzality a vazby mezi subjekty. Tímto se nad vrstvou reality vytváří vrstva modelů, ve které jsou prvky, třídy a jejich vztahy popsány metadaty. Tato metadata určují model a strukturu reálných dat.

V minulosti proběhlo velké množství projektů[125], jejichž cílem bylo vytvořit metamodely pro většinu používaných prostředků objektové analýzy a designu. Tyto metamodely měly být následně využity pro tvorbu jakéhosi univerzálního standardu (metametamodelu). Příkladem může být architektura COMMA (Common Object Methodology Metamodel Architecture), která je základem, jakýmsi podkladem, pro vytvoření univerzální metody objektové analýzy a designu.

10.3. Problém modelů a vznik meta-dat

Od vzniku modelování bylo vyvinuto několik rodin modelů, které obsahují specifické entity odpovídající potřebám modelů. Ve spojením s tím vznikly také robustní nástroje, které usnadňují správu těchto modelů. Časem i tyto robustní nástroje ztratily svou flexibilitu a schopnost přizpůsobit se změnám, což vedlo k omezení uživatelů, kteří museli pracovat s pevně definovanými vlastnostmi jednotlivých modelů, které neumožňovaly rozšíření.

Problém s nemožností provedení změň a rozšíření vedl k tomu, že bylo zapotřebí použít určitou úroveň abstrakce, tedy k přechodu k metamodelovanému náhledu - ke vzniku meta-metadat. Pomocí meta-metadat bylo dosaženo možnosti definovat a přizpůsobit jednotlivé modely. Metamodelování tedy vedlo ke tvorbě nových metodoologií pro tvorbu a návrh informačních systémů pomocí generických modelů.

Nedocenitelnou výhodou metamodelování je nezávislost na technologiích. Z tohoto důvodu se metamodely často využívají jako nadstavba a pro integraci různých modelačních technologií a modelačních jazyků. Tím dochází k překlenutí mezer mezi jednotlivými navrhovanými systémy.

Metamodely se liší stejně tak jako modely úhlem pohledu na danou problematiku díky specifickým potřebám vývojářů a implementátorů. Významný dopad na kvalitu metamodelu mají znalosti, zkušenosti a intuice zainteresovaných pracovníků, což může být leckdy velkým a občas i nepřekonatelným problémem.

[125] Smil, P. Úvod do tvorby IS s použitím objektového přístupu. Softwarové noviny, prosinec 1999, cis.12, ISSN: 1210-8472

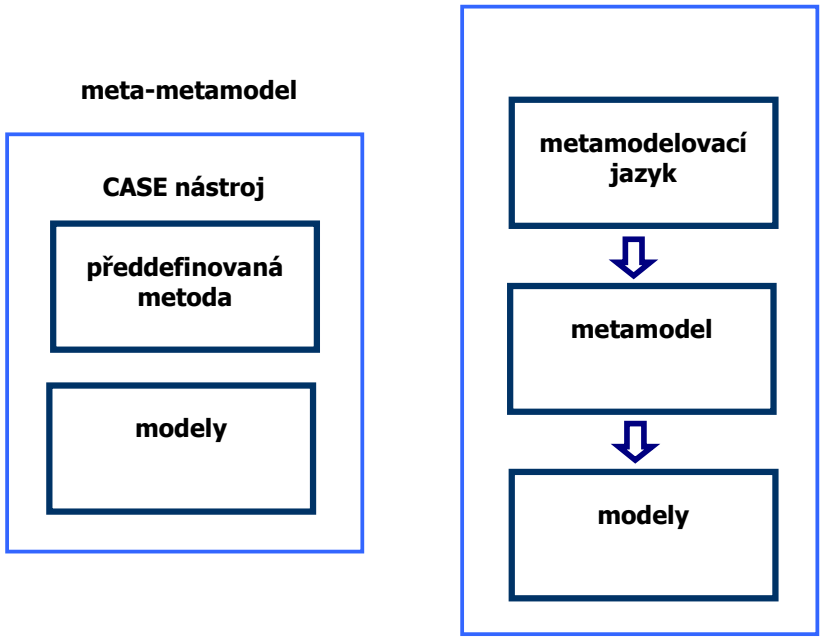
10.4. Využívání metamodelů

Metamodely se používají především:

- jako schémata pro sémantická data, která je zapotřebí uložit či komunikovat dále,
- pro jazyk, který podporuje konkrétní schéma či proces,
- pro jazyk, který doplňuje sémantiku k již existujícím informacím.

Jak bylo uvedeno výše, metamodelu se využívá jako nástroje pro definici modelu. Tedy jeho použití je především při vývoji nástrojů, které se zabývají modelováním - CASE nástrojů. Tyto nástroje využívají metamodelovací přístup pro rozšíření své funkčnosti o možnost úprav jednotlivých metod. Tyto metody se používají pro definici metadat, která tvoří model. Jak je patrné z níže uvedeného obrázku, ve srovnání s běžnými CASE nástroji, jejichž funkcionalita postihuje dvě nejnižší vrstvy modelu, jsou tyto nástroje rozšířeny na vrstvy tři (čtyři pro pohled z úrovně meta-metamodelu).

Obr. 23: Srovnání CASE a MetaCASE[126]



MetaCASE se liší od doposud uváděné metodologie pohledem na metamodelování pomocí CASE nástrojů s určitou úrovní abstrakce. Jedním z jazyků, popisujícím metamodely je podle ISO3 standard MOF[127] (Meta Object

[126] Balogh, I. Jankó, G. Murín, J. Žilka, R. Nástroje meta-CASE (charakteristika, prehled trhu, trendy) [online]. VŠE 2005. Dostupné na http://panrepa.org/CASE/meta_CASE.pdf. str. 4

[127] Object Management Group. Meta Object Facility [online]. Dostupné na: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=32622

Facility), definovaný skupinou OMG (Object Management Group) v normě ISO 19502:2005.

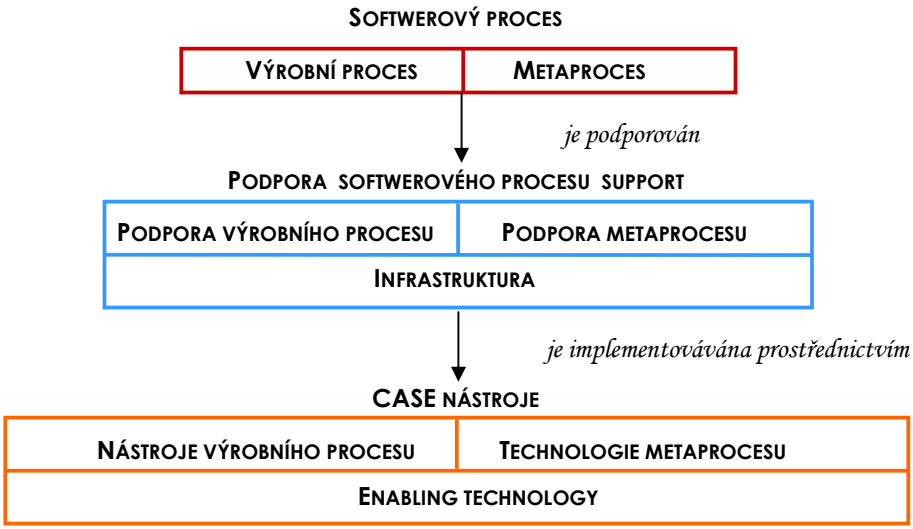
MOF jako jazyk specifikuje, vytváří a upravuje práci s technologií neurálních metamodelů, zajišťuje konzistenci mezi jednotlivými sadami modelů, implementuje design a použití repository aplikačních vývojových nástrojů atd. Repositoř se tedy stává úložištěm metadat (resp. jimi popsaných modelů), která za pomocí pravidel definovaných obdobnými metadaty hlídá svou integritu, resp. integritu modelů popsaných metadaty v ní obsažené. MOF je objektově orientovaný jazyk, který popisuje sám sebe, díky čemuž je vysoce modifikovatelný a rozšiřitelný. Současně definuje komunikační rozhraní, používaná pro správu a operace s modely, jejichž rozhraní dosud nebyla použita; k tomu používá UDK (Interface Definition Language) CORBA. Typickými metamodely (metamodelovacími jazyky) navrženými OMG jsou UML, Sysel, SPEM nebo CWM.

10.5. Všeobecný rámec softwarového inženýrství

Následující obrázek popisuje všeobecný rámec softwarového inženýrství přesněji zaměřeného na vývoj aplikačního softwaru.

Softwarový proces se skládá ze dvou podprocesů: výrobního procesu a metaprocesu. Produkční proces zahrnuje všechny aktivity, pravidla, metodologie, metody, organizační strukturu, nástroje, atd. které se používají pro formulaci, návrh, vývoj, předání a udržování softwarového produktu. Výrobní proces musí být definován, vyhodnocován a vyvíjen prostřednictvím metodického a systematického metaprocesu.

Obr. 24: Obecný rámec („the general framework“)[128]



Účelem metaprocesů je získávat a využívat nových produktů, které podporují všechny aktivity související s vývojem softwaru a to stále obecněji, tedy dosáhnout co nejvyšší úroveň obecnosti, dále usilují o zlepšení a inovace procesů, pravidel a technologií používané pro úspěšné dokončení systému.

Co se týče procesu výroby, můžeme mu poskytnout podporu či jej částečně zautomatizovat pomocí podpory výrobních nástrojů (production-process technology), neboť tak pomůžeme vývojářům specifikovat, vytvořit a provádět údržbu softwarového produktu. V organizaci bývají specifické nástroje a související procedury, směrnice a pokyny podporující výrobní proces nazývány proces podporující produkci (production-process support).

Metaproces může být automatizován a podporován dobře pomocí nástrojů metaprocesu (metaprocess technology), které jsou užívány pro vytvoření podpory pro metaprocesy (the metaprocess support). Podpora pro metaprocesy je specifická pomoc používaná v metaprocesech organizace k zautomatizování a navádění (ovlivňování) činností metaprocesu.

10.6. Závěr

V této kapitole jsme se zabývali metamodelováním a metainformačními systémy, neboť CASE nástroje se řadí mezi tzv. metasystémy.

Zaměřili jsme se na Metainformační systém MtS, který umožňuje popisovat, analyzovat a řídit IS/IT z pohledu všech významných dimenzí podle metodologie MMDIS z hlediska datového, funkčního a procesního,

[128] Fuggetta, A. Milano, P. CERFIEL a Classification of CASE Technology [online].The world's leading professional association for the advancement of technology 1993. Dostupné na: http://www.ieee.org/portal/site (navštíveno 5. dubna 2008). str 26

softwarového, hardwarového, organizačního, personálního, ekonomického, metodického, časového a z hlediska vzájemných vazeb těchto dimenzí.

Dále jsme se podívali na model nad model, tzv. metamodel, který vychází z popisu reality prostřednictvím modelu. Je to fungující systém, který se využívá při vývoji aplikací, informačních systémů a simulací (např. podnikových systémů). Samotný model je abstrakcí reality, která vytěsňuje nepotřebné a zbytečné informace a spojitosti.

Dále jsme se věnovali MetaCASE, který se liší od doposud uváděné metodologie pohledem na metamodelování pomocí CASE nástrojů s určitou úrovní abstrakce. Jedním z jazyků, popisujícím metamodely je podle ISO3 standard MOF (Meta Object Facility), definovaný skupinou OMG (Object Management Group) v normě ISO 19502:2005.

Závěrem jsme se zmínili o metaprocesu, jehož cílem je získávat a využívat nových produktů, které podporují všechny aktivity související s vývojem softwaru a to stále obecněji, tedy dosáhnout co nejvyšší úroveň obecnosti, dále usilují o zlepšení a inovace procesů, pravidel a technologií používané pro úspěšné dokončení systému.

11. Úroveň meta-metodiky metodického rámce

11.1. Úvod

V této kapitole se budeme věnovat metodickému rámci IS/ICT, principy pro vytvoření metodického vzoru. Dále se zaměříme na bázi znalostí metodického rámce a jak jej přizpůsobit dané firmě.

Pro pochopení následující kapitoly je třeba se nejprve zaměřit na vysvětlení pojmu MeFIS. MeFis[129] je „*metodický rámec (Methodology framework), tedy kolekce metodických vzorů pro různé domény, typy řešení a způsoby řešení spolu s principy a procesy pro vytvoření konkrétní metodiky.*" Tato kapitola slouží jako metodické východisko pro tvorbu vlastní metodiky pro výběr CASE nástroje.

11.2. Principy pro vytvoření metodického vzoru či metodiky

Pokud se chceme zabývat vytvořením metodického vzoru či konkrétní metodiky, je zapotřebí, abychom respektovali určité principy. Ty jsou v MeFIS označeny jako principy meta-metodiky. Autorka se zmiňuje o devíti principech pro meta-metodiky, které vycházejí z prací A. Cockburna a J. Highsithe:

1) Preference osobní komunikace před ostatními formami komunikace

Je dokázáno, že osobní komunikace dvou lidí je tou nejefektivnější formou komunikace a to z níže uvedených důvodů:

- fyzická blízkost (vizuální podněty, timing, 3dimenzionalita, vůně);
- různé způsoby komunikace (gesta, slova, mimika);
- hlasové zabarvení a časování;
- otázky a odpovědi v reálném čase, otázky signalizují nedostatečné vysvětlení nebo místo, kde chybí zázemí posluchače.

Cocburn uvádí, že nejméně efektivní je písemná komunikace, která se používá v metodikách budování informačních systému a informačních a komunikačních technologií a to jako základní nástroj komunikace.

Jak zlepšit metodiku? Vybrat si komunikaci pokud možno v co nejefektivnější formě, jako je např.: v nejlepším případě osobní komunikace

[129] Buchalcevová, A. Metodiky vývoje a údržby informačních systémů. Grada, Praha 2005, ISBN 80-247-1075-7. str. 154

v horším videokonference, telefon, elektronická pošta, videozáznam či audio záznam.

2) Váha metodiky[130] je přímo úměrná velikosti týmu

Větší tým vyžaduje složitější komunikaci, pokud je tedy metodika také určitým způsobem komunikace lidí a jejich koordinace, poté platí přímá úměrnost počtu lidí zainteresovaných na projektu s váhou metodiky.

- Nelze použít identickou metodiku pro malý i velký tým.

3) Relativně malé zvýšení váhy metodiky vede k relativně většímu objemu nákladů projektu

Jedná se o nejdůležitější princip, na kterém záleží úspěšnost projektu. Uvádí se zde, že pro malý tým je dostačující relativně „lehká" metodika, protože malému týmu umožňuje větší produktivitu a efektivnost než metodika „těžká." Pokud by malý tým pracoval s těžkou metodikou, mohlo by to jejich produktivitu a efektivnost naopak snížit, jelikož těžká metodika je přizpůsobena pro práci velkého týmu, kde jsou jednotlivé úkoly rozděleny mezi více lidí a je založena na komplikovanější koordinaci a komunikaci.

Pokud bych to měla více přiblížit, je to, jako by jedna osoba vyvíjela jednoduchý program, který zvládne namodelovat, naprogramovat, otestovat a dát do provozu během např. týdne pilné práce. Pokud tento člověk bude postupovat podle „lehké metodiky", tak si namodeluje funkcionalitu buď na papír nebo v lepším případě pomocí jednoduchého CASE nástroje, který mu ulehčí práci vygenerováním části kódu přímo z návrhu IS. Poté proběhne fáze programování, následuje testování chyb a jejich oprava a nakonec implementace hotové IS. Pokud by ale postupoval podle „těžké metodiky", poté by si musela rozdělit několik rolí jen pro svou jedinou měnící se roli. Zbytečně by musel provádět úkony, které jsou v metodice uvedeny a které jsou naprosto zbytečné (např. komunikace mezi jednotlivými rolemi).

A stejně tak je to i u malého a velkého týmu. Malý tým by zbytečně zdržovalo přidělení více rolí jednomu pracovníkovi, nadbytečná komunikace a koordinace a tím by činilo řízení IS mnohem složitější a komplikovanější a vedlo by spíše ke snížení jejich efektivnosti díky provádění zbytečných úkonů.

Z výše uvedeného vidíme, že při vedení projektu hraje klíčovou roli zvolení správné metodiky.

Autorka se zmiňuje o „hranici velikosti projektu," což je bod, ve kterém určitý počet lidí schopen mezi sebou komunikovat a vzájemně se koordinovat a přitom řešit určitý problém. Problém spočívá v tom, že je velmi těžké (a často i nemožné) určit hned na začátku velikost problému a počet lidí, kteří se budou zabývat jeho řešením. Uvádí se, že je tato hranice vyšší pro velký tým s „těžkou metodikou" a nižší pro malý tým s „lehčí" metodikou.

[130] „Váha metodiky" je kritérium metodiky, podle kterého jsou metodiky členěny na těžké a lehké.

4) Projekty větší důležitosti vyžadují „těžší“[131] metodiky

Tento princip bývá při vývoji IS samozřejmý. Vyplývá obvykle z toho, že čím je projekt důležitější, tím vyšší rozpočet je ochoten jeho zadavatel pro vývoj IS vynaložit, i když sám pojem důležitost je velice relativní, neboť, co je pro jednu osobu nadměrně důležité, může být pro druhou naprosto nepodstatné a nedůležité. Proto také dochází k situacím, kdy na naprosto běžném projektu pracuje na náš vkus zbytečně mnoho lidí a projekt je tak zbytečně naddimenzován.

Autorka každopádně uvádí, že „čím je vytvářený systém důležitější pro organizaci, tím „*těžší*“ *musí být metodika pro jeho vytvoření i za cenu vyšších nákladů.*“

Je tedy otázkou, z jaké důležitosti by se při vývoji IS mělo vycházet, zda z důležitosti, kterou považuje klient či z důležitosti, kterou považují samotní pracovníci zúčastnění na vývoji IS.

5) Disciplína není formalita, proces není dovednost, dokumentace není pochopení

Autorka zde uvádí, že jen 15-20% pochopení, které je třeba pro vývoj, je dosaženo pomocí dokumentu Specifikace požadavků. Proto je třeba, aby vyššího pochopení bylo dosaženo komunikací mezi lidmi. Dále uvádí, že disciplína je nezbytná, ale může být i neformální. Je to z důvodu[132], že ne všechny naše znalosti a dovednosti je možné určitým způsobem zaznamenat v rámci určitého procesu, neboť určité znalosti a dovednosti jsou člověkem vžity až během praxe a nelze je popsat v rámci určitého postupu (což je všeobecný problém oblasti knowledge managementu). Toto byl mimo jiné důvod pro vznik agilních přístupů, neboť „*sebedokonalejší přístupy nenaradí dovednosti a kvalifikaci lidí.*“

6) Zvýšení zpětné vazby a komunikace snižuje potřebu meziproduktů

Nejčastějším důvodem, který vyvolává potřebu meziproduktu, je špatná komunikace a koordinace. Je tedy zapotřebí neopomínat důležitost komunikace a podpořit ji co nejvíce, neboť modely a psaná dokumentace nejsou dostačující.

7) Činnosti, které nejsou úzkým místem, nemusí být nutně efektivní svázat s následujícím

Podle teorie omezení není vždy nutné vyhledat úzké místo a to se snažit rozšířit. V některých případech je neefektivnost tohoto úzkého místa efektivní neefektivnost je efektivní? pro průběh celého procesu, naopak jeho efektivnost ještě zvyšuje.

---

[131] Těžké mají podrobný popis a jsou rozpracovány na detailní úrovni. Lehké metodiky jsou metodiky, u kterých stačí, aby minimálně dostačovaly (Cockburn: "barely sufficient", Highsmith: "a little bit less than just enough").

[132] Hradecká, P. Knowledge management. Diplomová práce, VŠE Praha 2006. str. 58

8) Zaměření na hodnotu pro zákazníka

Jde o to, zaměřit se pouze na core procesy.

9) Možné vyžádání větší formality metodiky

Může nastat případ, kdy je požadována větší formálnost metodiky a to z důvodů právních či bezpečnostních.

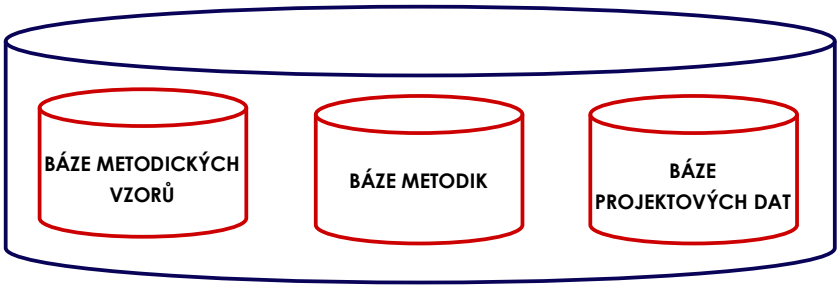
11.3. Báze znalostí a přizpůsobení metodického rámce

Obr. 25: Metodický rámec MeFIS[133]

MeFIS je tedy tzv. meta-metodikou pro vývoj IS/ICT, neboť představuje ucelený rámec metodických přístupů v bázi metodických vzorů, kterou je zapotřebí neustále rozvíjet a udržovat.

MeFis obsahuje bázi znalostí, kterou tvoří tři druhy bází: báze metodik, báze projektových dat a báze metodických vzorů. Báze projektových dat obsahuje data o konkrétních projektech, která jsou velmi užitečná při tvorbě analýzy, odhadech rozpočtu, času a pracnosti budoucích projektů.

Obr. 26: Báze znalostí MeFIS[134]

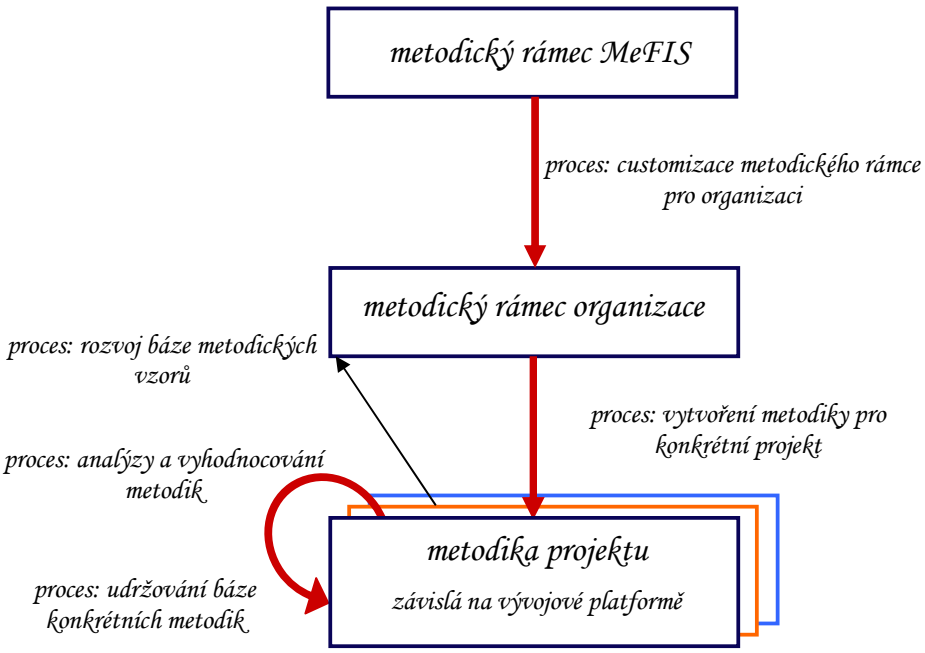


MeFis obsahuje bázi znalostí, kterou tvoří tři druhy bází: báze metodik, báze projektových dat a báze metodických vzorů. Báze projektových dat obsahuje data o konkrétních projektech, která jsou velmi užitečná při tvorbě analýzy, odhadech rozpočtu, času a pracnosti budoucích projektů.

[133] Buchalcevová, A. Metodiky vývoje a údržby informačních systémů. Grada, Praha 2005, ISBN 80-247-1075-7. str. 12

[134] Buchalcevová, A. Metodiky vývoje a údržby informačních systémů. Grada, Praha 2005, ISBN 80-247-1075-7. str. 10

Obr. 27: Procesy meta-metodiky[135]



Z důvodu jedinečnosti každé firmy, je třeba metodický rámec přizpůsobit na míru podmínkám organizace. Toto přizpůsobení probíhá na základě:

- firemní kultury,
- core procesy,
- organizační strukturou,
- standardy, best practices
- nástroji, které se v organizaci používají,
- metodikami, které se v organizace používají.

Otázkou je, zda se někdy firma tolik nevymyká, že je zapotřebí přizpůsobit určitou z uvedených podmínek metodickému rámci. Např. v oblasti nástrojů a metodik, které organizace používá, je nutno v určitých případech provést změny. Proto bych nástroje a metodiky nepovažovala za klíčové podmínky, ale zvážila bych zde možnost případné změny. Jak uvádí sama autorka, nejvíce ovlivňuje customizace firemní kultura a ostatní faktory se uplatňují spíše výjimečně. Podle mého názoru bychom měli zohlednit především core procesy, dále pak best practices a standardy.

Nakonec si ještě uvedeme, co zahrnuje přizpůsobení na základě firemní kultury:

---

[135] Buchalcevová, A. Metodiky vývoje a údržby informačních systémů. Grada, Praha 2005, ISBN 80-247-1075-7. str. 100

- modifikaci základních principů metodického vzoru;
- výběr metodických vzorů z báze lehkých a těžkých metodik;
- úpravu procesů v rámci metodických vzorů;
- úpravu hodnotících kritérií v rámci bran na konci fází;
- stanovení odlišných metrik;
- stanovení standardů;
- definici nových vzorů.

11.4. Závěr

V této kapitole jsme se zabývali metodickým rámcem známým pod zkratkou MeFIS, který je kolekcí metodických vzorů pro různé domény, typy řešení a způsoby řešení spolu s principy a procesy pro vytvoření konkrétní metodiky. Jedná se o tzv. meta-metodika pro vývoj IS/ICT, neboť představuje ucelený metodických přístupů v bázi metodických vzorů, kterou je zapotřebí neustále rozvíjet a udržovat. MeFIS obsahuje tři druhy bází: metodických vzorů, metodik a projektových dat.

Uvedli jsme si principy pro vytvoření metodického vzoru a blíže jsme se podívali na báze znalostí metodického rámce a na oblasti, ve kterých je třeba jej přizpůsobit podmínkám dané organizace a firemní kultury.

12. Aktuální situace na trhu CASE nástrojů

Před samotnou problematikou výběru vhodného CASE nástroje pro určitou firmu si nejprve utvořme přehled o situaci na trhu s CASE nástroji. Tato kapitola by měla sloužit jako úvod do problematiky praktického použití CASE nástrojů a jako představení jejich druhů a schopností. Na základě těchto znalostí pak můžeme mnohem lépe porozumět návrhu metodiky, která byla pro pokrytí stávajícího stavu navržena.

Tato kapitola je rozdělena na dvě části. V první části se zabývám popisem jednotlivých nástrojů s přihlédnutím k jejich unikátním vlastnostem nebo k faktům, které jsou dle mého názoru důležité pro potenciálního zájemce, v druhé části provádím přímé srovnání těchto nástrojů ve formě objektivního posouzení jejich vlastností bez hodnocení jejich přínosu – metodou je srovnávací tabulka.

Stávající situace na trhu s modelovacími nástroji se odvíjí od situace na obecném trhu se software. Jednou z jeho vlastností je jeho globalizace. To se odvíjí od toho, že software je veskrze věc nehmotná, a pokud se na něj podíváme jako na shluk dat, i libovolně přenositelná. Lze jej tedy distribuovat téměř libovolně nezávisle na geografických podmínkách. Tomu napomáhá také jeho propagace na firemních stánkách apod. pomocí internetu, která díky tomu také boří omezení známě z běžného světa jako jsou regionální hranice. Jazyková bariéra je v oblasti korporátního software často chápána až jako druhotná podmínka při posuzování použitelnosti daného nástroje, protože na rozdíl od spotřebitelského sektoru domácností je v korporátní sféře počet lingvisticky zdatných uživatelů prakticky stoprocentní.I přesto ale přítomnost české jazykové mutace v nástroji beru jako jeden ze sledovaných ukazatelů, protože tato vlastnost může být pro výběr nástroje dostatečně významná a vytvořit prostor pro konkurenční výhodu výrobku.

Stávající trh s CASE nástroji lze rozdělit podle několika možných hledisek. Vyjma toho, zda nástroj obsahuje základní předpokládané funkce, jako je podpora generování různých typů diagramů, podpora kontroly těchto diagramů, podpora různých formátů vstupů a výstupů, podpora kooperativní práce, je pro mnoho firem důležitá především jeho cena.I když sama v rámci navrhované metodiky uvádím cenu až jako druhotné srovnávací hledisko, v tomto přehledu uvádím u každého nástroje přehled aktuálních cen dle aktuálních ceníků uveřejněných na internetových stránkách tvůrců nebo prodejců (referenční datum 15.6.2008).

Pojďme si nyní představit vybrané představitele CASE nástrojů přítomných na našem trhu.

12.1. Microsoft Visio

Aktuální verze: Microsoft Office Visio 2007

Výrobce: Microsoft Corporation

Distributor: v ČR Microsoft s.r.o.

Cena licence:

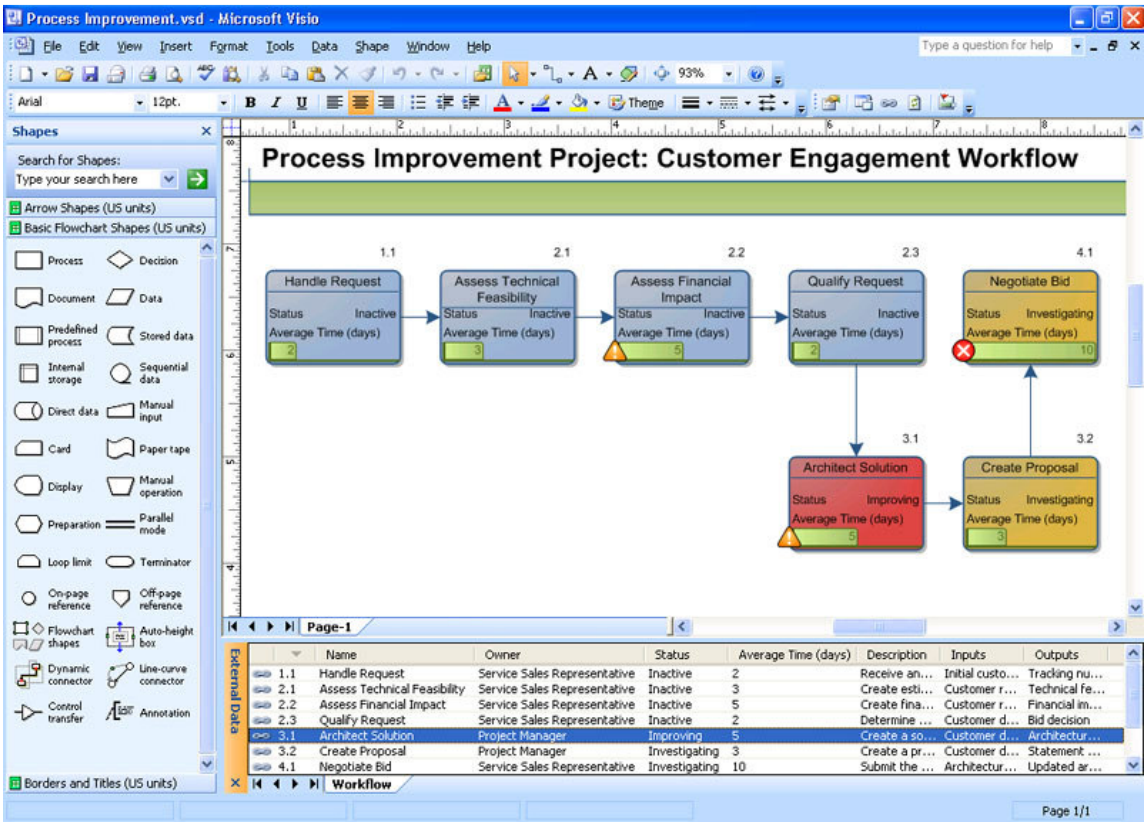
- · Standard edition : 7330Kč / 3560Kč (Nový SW / Upgrade) / jeden počítač = uživatel
- · Professional edition : 16090Kč / 9980Kč (Nový SW / Upgrade) / jeden počítač = uživatel

Lokalizace do češtiny: Ano

Demoverze: Ano (60 dní)

Platforma: Microsoft Windows (XP, Vista)

Obr. 28: Microsoft Visio[136]



Podpora diagramů: vývojové diagramy, DFD, BPM, schémata pro BI, datové modely, diagramy sítí, časové osy a kalendáře, organizační schémata, workflow diagramy, diagramy definované standardem UML

Nástroj Microsoft Visio je v současné době součástí produktu Microsoft Office, s nímž je i distribuován ve verzi s přívlastkem Visio. Jedná se o prvek integrovaný do tohoto kancelářského systému.

[136] http://www.microsoft.com/cze/office/programs/visio/overview.mspx

Vzhledem k tomu, že se jedná o relativně jednoduchý nástroj, bez analýzy a kontroly stavu modelů, nehodí se do větších týmů nebo na větší projekty. Sdílení projektů probíhá pomocí technologie Windows SharePoint Services, která umožňuje offline sdílet dokumenty a soubory v prostředí uzavřených skupin. U nástroje naopak překvapuje velmi dobré uživatelské rozhraní.

V oblasti propojení na další analytické nástroje lze říci, že je podporován pouze formát XML, samozřejmě lze výsledky přenášet v rámci programů Microsoft Office. Ve verzi Professional je nástroj schopen výsledný model exportovat jako sekvenci příkazů pro tvorbu databáze přes propojení ODBC do MS SQL Server, MS SharePoint server a další servery s tímto rozhraním.

Nástroj je schopný provádět omezený reverzní engineering z formátu MS VisualBasic 6, VisualBasic.NET, Visual C++ a Visual C#. Mimo efektně zvládnutou podporu kreslení diagramů ale Microsoft Visio nemá podporu ani pokročilých funkcí CASE nástrojů, přičemž například nástroj Dia, který bude představen později, disponuje téměř stejnými možnostmi, ale je distribuován zdarma.

Výše uvedené informace pocházejí z webových stránek výrobce.[137]

12.2. Oracle Designer

Aktuální verze: Oracle Designer 10g Rel2 10.1.2.4

Výrobce: Oracle Corp.

Distributor: Oracle Czech, s.r.o.

Cena licence:

- Named User Plus (licence na jednoho uživatele za celý balík Oracle Developer Suite včetně podpory 1 rok) – 6500 USD
- Update + podpora 1 rok – 1400USD
- Při použití OTN licence (při akceptování jejích omezení) lze získat produkty firmy Oracle zadarmo

Lokalizace do češtiny: Ne

Demoverze: Není, ale lze si stáhnout zadarmo neomezenou licenci z webu při OTN licenci.

Platforma: Microsoft Windows (XP, 2000, Server 2003), Linux, Solaris

Podpora diagramů: Procesní model, Datové modely, Diagram datových toků, Diagram hierarchií podnikových funkcí, Konceptuální datový model, Klasický datový model, Multidimensionální model

Oracle Designer je dodáván jako součást balíčku Oracle Developer Suite – díky tomu patří do rodiny velice robustních nástrojů. Funkčnost tohoto nástroje se člení do čtyř relativně samostatných skupin nástrojů, které zhruba kopírují potřeby v jednotlivých fázích vývoje systému – nástroje pro modelování

[137] http://www.microsoft.com/cze/office/programs/visio/highlights.mspx

systémů, nástroje pro transformaci předběžných návrhů, prostředí pro návrh a generování a poslední částí je samotná repository.

Oracle designer je úzce svázaný s databázovým systémem Oracle, do kterého je schopen přímo exportovat modelované aplikace. Navíc oplývá unikátní schopností modelování klient/server aplikací s podporou webových portálů. Modelování do výše uvedených diagramů a jejich ukládání do repository modelů je prováděno inkrementálně, díky čemuž lze modely verzovat, samozřejmá je pak podpora kooperace vývojářů.–

Díky výše zmíněné úzké svázanosti s DB Oracle je schopen podpory evidence triggerů, procedur a views, spolupracuje s bezpečnostní politikou SŘBD. Je schopen generovat výstupy v jazycích C++, Visual Basic, Java.

Vysoká cena nástroje omezuje jeho použití (mimo omezení OTN licence) na finančně zajištěné organizace.

Výše uvedené informace pocházejí z webových stránek výrobce.[138]

12.3. Select Architect

Aktuální verze: Select Solution Factory 7.0

Výrobce: Select Business Solutions

Distributor: Select Business Solutions, v ČR LBMS – www.lbms.cz

Cena licence:

- 1 uživatel (na jméno) – 88050 Kč
- 2-4 uživatelé (souběžný přístup) – 132075 Kč

Lokalizace do češtiny: Ne

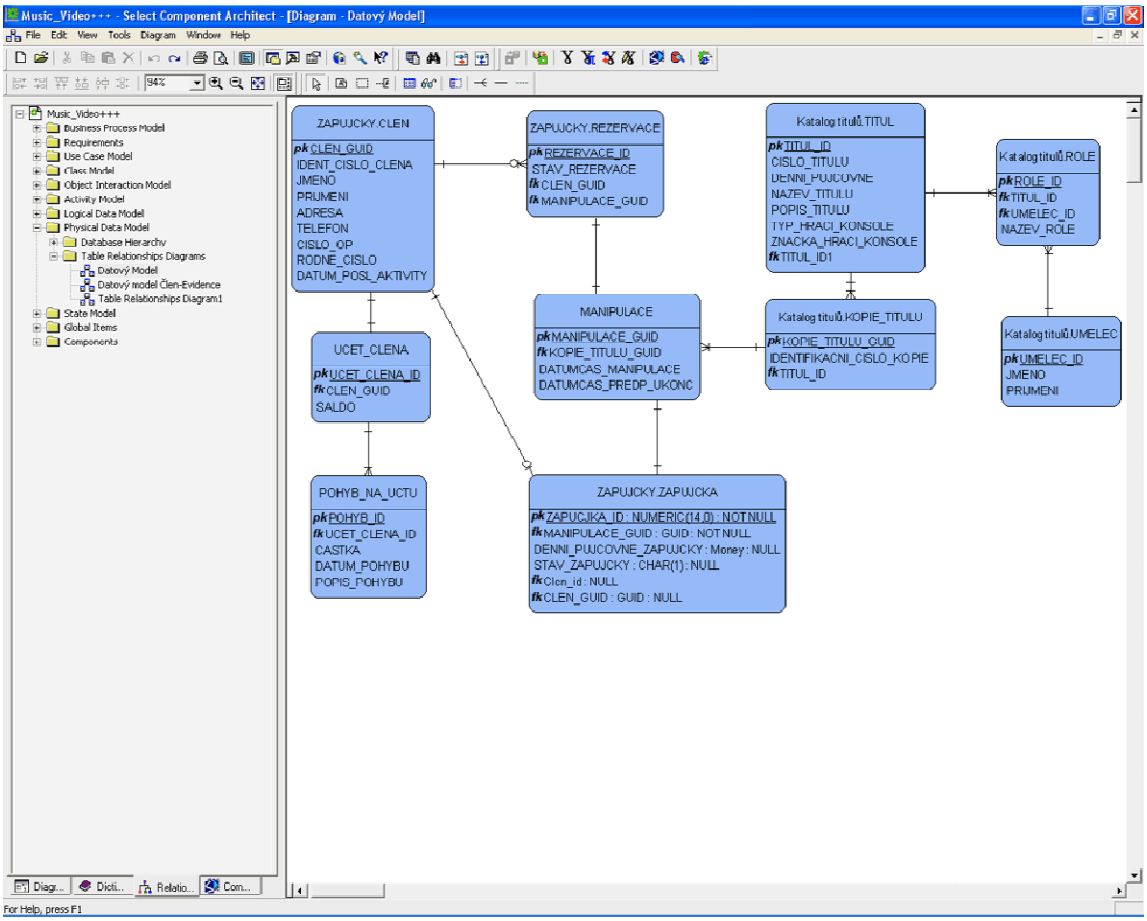
Demoverze: Ano

Platforma: Microsoft Windows (XP, 2000), Linux, Solaris

Podpora diagramů: Business Process Model, Use Case diagram, diagram tříd, Object collaboration diagram, State transition diagram, sekvenční diagram, Procesní hierarchický diagram, Process Thread diagram, relační model

[138] http://www.oracle.com/technology/products/designer/index.html, http://www.oracle.com/technology/products/designer/demos.htm, http://www.oracle.com/technology/products/designer/documentation.html

Obr. 29: Select Architect[139]



Select Architect je jako nástroj spolu s ostatními nástroji v rámci dodávaného balíku vhodný pro vytváření aplikací vícevrstevné architektury. Select Architect má jako jeden z mála modelovacích nástrojů na trhu vlastní unikátní modelovací metodiku - CA Process Continuum. Tato metodika je aplikací metodiky LBMS Development Method, podporuje objektový model a současně ERA model, který je využíván pro ukládání stavů objektů do relační DB.

Díky zakomponované kooperaci s databázemi, je příjemnou vlastností evidence databázových objektů a přístupových práv k tabulkám, nechybí také generování databázových schémat jako součást standardního výstupu.

Výstup do programovacího kódu je nástrojem podporován do standardní kolekce jazyků jako je C++, Java, Visual Basic, Corba IDL, zajímavá je podpora dnes méně užívaného jazyka Pascal v prostředí Delphi. K databázím nástroj přistupuje pomocí pluginů nebo nativně pomocí přímého přístupu (tedy lze přímo editovat jednotlivé komponenty SŘBD). Mezi podporované systémy patří MS SQL Server, Oracle, Informix, Sybase, DB/2, a Interbase, přičemž tento výčet není konečný. Výstupem může být také XML.

---

[139] http://www.lbms.cz/Nastroje/Select-Architect/uzivatelske-rozhrani.htm

Nástroj podporuje kooperační práci na projektu pomocí systémového serveru, který podporuje práci několika desítek souběžně pracujících uživatelů, to vše s plně funkčním systémem přístupových práv k uchovávaným informacím. Repository, ve které jsou ukládány modely, je čistě objektová SOFTLAB.

Nástroj dokáže velmi dobře spolupracovat s partnerskými produkty Eclipse 3.1, LogicWorks Erwin, Mercury Interactive Test Director, podporuje generování manuálů ve formě MS Word dokumentu.

Uživatelský komfort a kooperaci při řešení zvyšuje zajímavá vlastnost, kdy nástroj umožňuje vyjma centrálního sdílenému přístupu k jednotlivým modelům také to, že tyto modely mohou jednotliví uživatelé nezávisle na sobě paralelně verzovat, tedy nejsou navzájem ovlivnitelní.

Možnosti modifikace a kastomizace nástroje jsou slabší, je možné upravit v nástroji například vzhled šablon generovaných dokumentů a vizualizaci objektů.

Vzhledem k ceně bych tento produkt stejně jako nástroj firmy Oracle doporučila spíše pro větší podniky s dostačeným finančním zázemím. Select Architect je dodáván v balíku s množinou dalších komponent pro vytváření aplikací vícevrstvé architektury, je zcela dostačujícím a plně vybaveným systémem pro vývoj aplikací.

12.4. IBM Rational ROSE Software Modeler / Data Architect

Aktuální verze: 7.0.5

Výrobce: IBM

Distributor: v ČR více firem

Cena licence: variabilní, pohybuje se v rozmezí od 2.650 US za verzi Developer for Java Authorised User License, po 11.100 US za Technical Developer Floating User License

Licence jsou dvojího druhu - Authorised User License – licence je nepřenosná, kupuje se na jednotlivého uživatele, Floating User License - přenosná, přičemž se hlídá maximální počet souběžně přistupujících uživatelů k programu

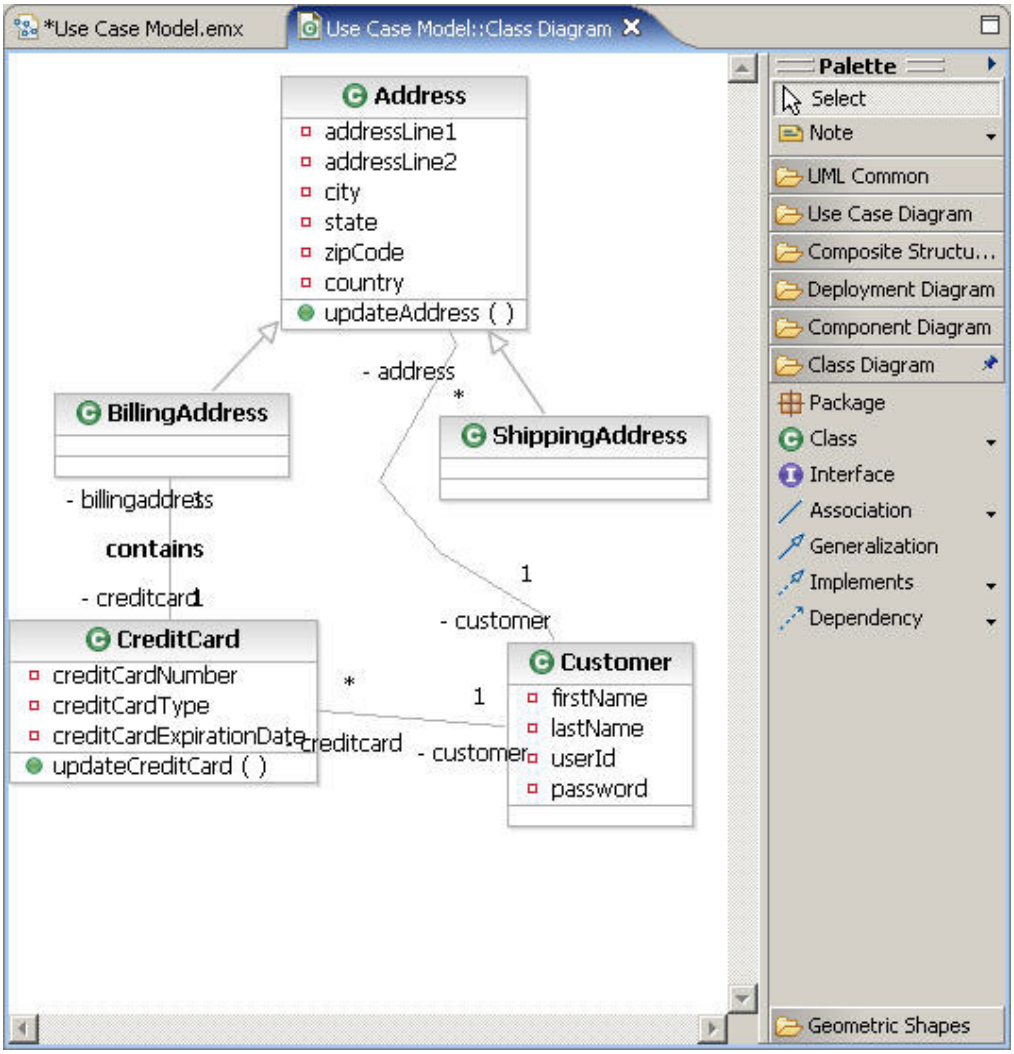
Lokalizace do češtiny: Ne.

Demoverze: Ano, 30ti denní

Platforma: Windows, Unixová platforma

Podpora diagramů: diagramy UML - use case diagram, sekvenční diagram, diagram spolupráce, diagram tříd, stavový diagram, diagram komponent, diagram nasazení, diagram aktivit

Obr. 30: IBM Rational Software modeler[140]



IBM Rational Software modeler je jako jeden z mála představitelem vlastní používané metodiky, kterou podporuje. Jedná se o interní metodiku RUP (viz. kapitola 58.5.3).

V několika posledních verzích byl nástroj Rational Software Modeler nově rozšířen o výraznou podporu Web services, SOA a UML 2.x. V rámci designu je možné použít různé návrhové vzory, modely je možné mezi sebou transformovat a porovnávat, případně spojovat či rozdělovat. Uložení do interní repository, vycházející ze systému Rational ClearCase, umožňuje verzování a podporu kooperativního zpracování, které je řešeno obvyklým příznakovým zamykáním dokumentů. V případě potřeby je nástroj schopný rozšířit o verzovací nástroj MS SourceSafe.

[140] http://www-128.ibm.com/developerworks/rational/library/05/329_kunal/

Výhodou pro širší možnosti kastomizace je, že nástroj podporuje metamodelování, frameworky projektů pro jednotlivé projektové týmy, skriptovací jazyk pro úpravu modelů RoseScript a tzv. stereotypy, což je rozšíření, které umožňuje použití existujících prvků UML pro vytvoření prvků nových. Bez zajímavosti není, že program využívá rozhraní IDE Eclipse ve formě rozšiřitelného modulu.

Výstup programu je možný do několika programovacích jazyků, nativní podpora je pro jazyky Java a její rozšíření o standard J2EE, C, ANSI C++, Ada 83, Ada 85, Java, Visual Basic, XML přičemž nástroj obsahuje pluginy pro vývojová prostředí Borland JBuilder, MS Visual Studio, Eclipse.

Z hlediska podpory databázového propojení je Rational Software Modeler schopen komunikovat s SŘBD DB2, MS SQL Server, Oracle, Sybase a ANSI SQL 92. Z hlediska propojení s ostatními nástroji je Rational Software Modeler primárně zaměřen na ostatní nástroje IBM Rose – jde například o Rational Web Developer, Rational Application Developer, Rational Software Architect, Rational Data Modeler.

Vyšší cenu firma IBM odůvodňuje kvalitní zákaznickou podporou, ale i tak je Rational Software Modeler spíše vhodný do větších firem s širokým portfoliem modelovacích potřeb, kde díky široké knihovně modelů a podpoře metamodelování, najde své uplatnění.

Výše uvedené informace pocházejí z webových stránek výrobce.[141]

12.5. **Sybase Power** Designer

SYBASE

Aktuální verze: Sybase Power Designer 12.5

Výrobce: Sybase Inc.

Distributor: Sybase Software, s.r.o.

Cena licence:

- PowerDesigner Data Architect 12.5 - $2995.00
- PowerDesigner Data Architect Enterprise 12.5 - $4990.00
- PowerDesigner Developer 12.5 - $2995.00
- PowerDesigner Developer Enterprise 12.5 - $4990.00
- PowerDesigner Studio 12.5 - $5995.00
- PowerDesginer Studio Enterprise 12.5 – $7495.00

Demoverze: Ano, 15 dní

Lokalizace do češtiny: Ne.

---

[141] http://www.selectbs.com/products/select-architect.htm, http://www.selectbs.com/downloads/products/Select-Architect.pdf

Platforma: MS Windows

Podpora diagramů: konceptuální, logické a fyzické datové modely, Data Warehouse modely, Use Case Daigram, diagram tříd, diagram interakcí, Statechart diagram, diagram aktivit, sekvenční graf

Power Designer podporuje objektové modelování standardu UML 2.0, dále má implementovány standardy BPEL4WS, BPMN, DTD, ebXML, IDEF, RDBMS. Rozsah problematiky zpracovávané tímto softwarem je široký – od procesní analýzy podniku po datovou a objektovou analýzu informačních systémů. Všechny tyto modely lze převádět navzájem mezi sebou, díky čemuž lze dostat komplexní a ucelený pohled na modelovanou problematiku.

Nástroj pracuje primárně s vlastní, velmi dobře zpracovanou, repository, která podporuje verzování a plně nastavitelná uživatelská oprávnění. Nástroj je schopný generovat z modelů kódy pro jazyky Java, C#, VB .NET, Hibernate, EJB3, NHibernate, JSF, WinForm (.NET a .NET CF) a pak pomocí sesterského nástroje Sybase Power Builder s nimi dále pracovat. Mezi výstupy se také řadí podpora XML, XML Schema a DTD standardů Nástroj se možno přidat jako plugin do nástroje Eclipse nebo jako addin od MS Visual Studio, s kterými je relativně těsně svázaný. S pomocí tohoto nástroje je možné na základě modelů generovat reporty například ve formátu RTF nebo HTML a vytvářet analýzy potřeb.

Takto robustní nástroj je především využitelný ve velkých firmách, kde je schopen pokrýt i ty nejnáročnější požadavky, což obnáší zvýšené náklady na jeho pořízení a podporu. Verze nástroje bez přídomku Enterprise se dodává bez repository, což může být vhodné řešení menším týmům.

Výše uvedené informace pocházejí z webových stránek výrobce.[142]

12.6. Altova UModel

Aktuální verze: Altova UModel 2008

Výrobce: Altova

Distributor:

Cena licence: za 1 licenci s roční podporou, jinak množstevní sleva

enterpise edition – 149€

professional edition – 99€

Demoverze: Ano, 30 dní

Lokalizace do češtiny: Ne.

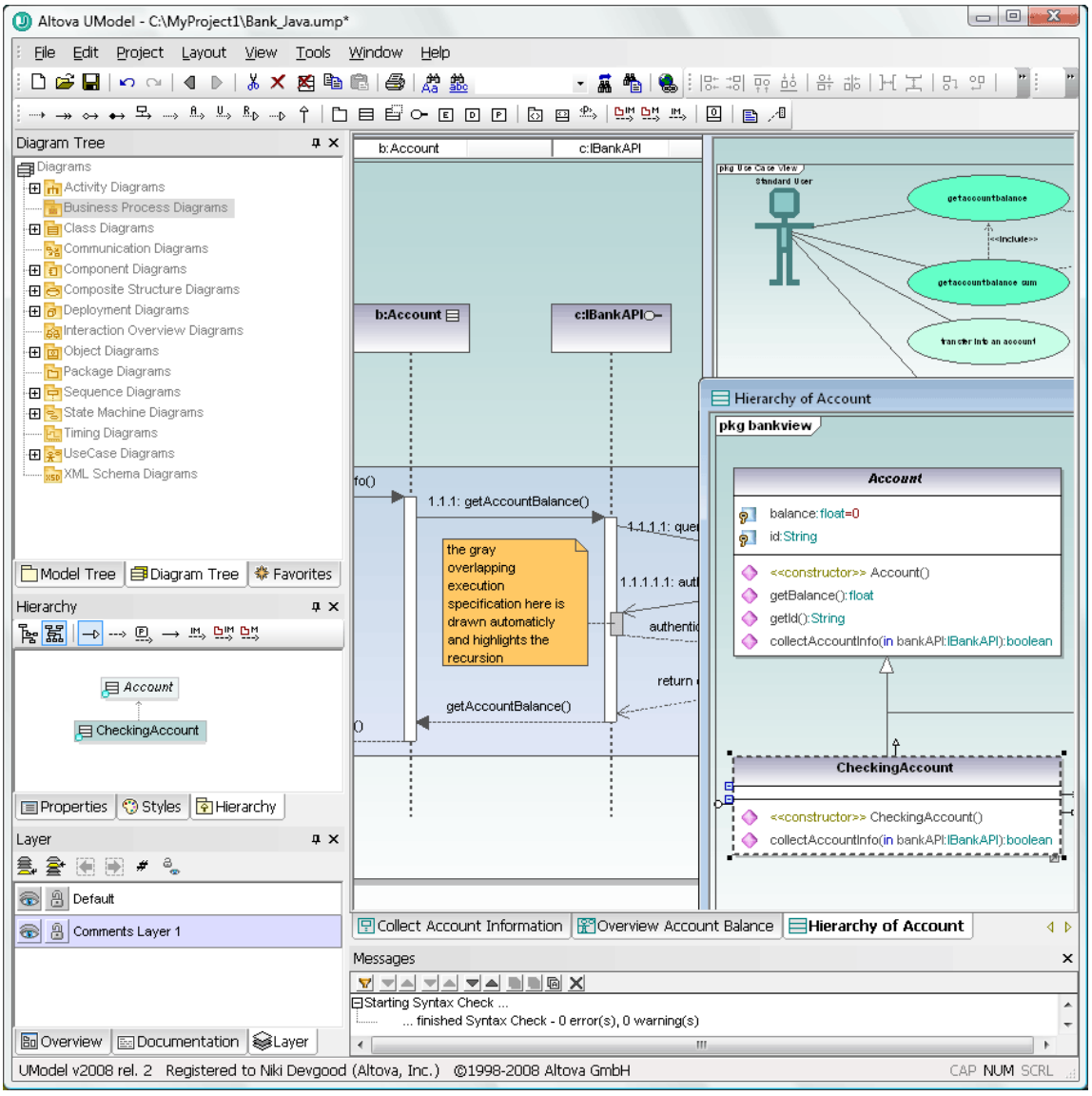
Platforma: Windows (NT 4.0, 2000, XP, Server 2003)

Podpora diagramů: Use Case Diagram, diagram tříd, diagram komponent, diagram nasazení, sekvenční diagram, časový diagram (chronogram), Business

[142] http://www.sybase.cz/buxus/generate_page.php?page_id=1

Process Diagram, diagram aktivit, komunikační diagram, package diagram, State Machine Diagram, XML Schema jako UML diagram

Obr. 31: Altova UModel[143]



Altova UModel se nehlásí k robustním systémům schopným celopodnikové integrace, přesto výrazným způsobem zaujme. Nástroj vychází z UML 2.1.2 a doplňuje jej o XML schema a BPMN diagramy. Jedná se o graficky přehledný, snadno ovladatelný program, jehož lze využít po všechny fáze vývoje produktu.

Altova UModel dovoluje on-demand generování zdrojových kódů Javy, C# a VB.NET, přičemž pro stejné kódy podporuje i reverzní engineering. Další

[143] http://www.altova.com/products/umodel/uml_tool.html

možností je podpora MS Virtual Studio a Eclipse IDE, real-time synchronizace UML modelů s programovými kódy.

Altova UModel podporuje sdílení projektů, resp. jejich součástí mezi jednotlivými projekty pomocí tzv. sdílených balíčků (Share Packets), do kterého je možné vložit objekty, metody a další prvky diagramů. Sdílení modelů v repository není nativně podporováno, musí se řešit aplikacemi třetích stran.

Nástroj je svázaný s dalšími výrobky firmy Altova, například nabízí nativní kooperaci s prohlížečem XML souborů XML Spy, podpora XML je vyšší než u jiných CASE nástrojů – příkladem může být XML Schema jako UML diagram.

Vzhledem k relativně nízké ceně, podpoře nejmodernějších objektových modelovacích standardů, rozsahu podpory nástroje a jednoduchosti ovládání, jej řadím ke špičce mezi dostupnými nástroji pro menší i větší podniky, pokud jejich výběr není výrazně omezen specifickou potřebou.

Výše uvedené informace pocházejí z webových stránek výrobce.[144]

12.7. Dia

Aktuální verze: 0.96.1

Výrobce: -, opensource nástroj

Distributor: -, je distribuován pomocí internetu na http://live.gnome.org/Dia

Cena licence: -, ke stažení a používání zdarma

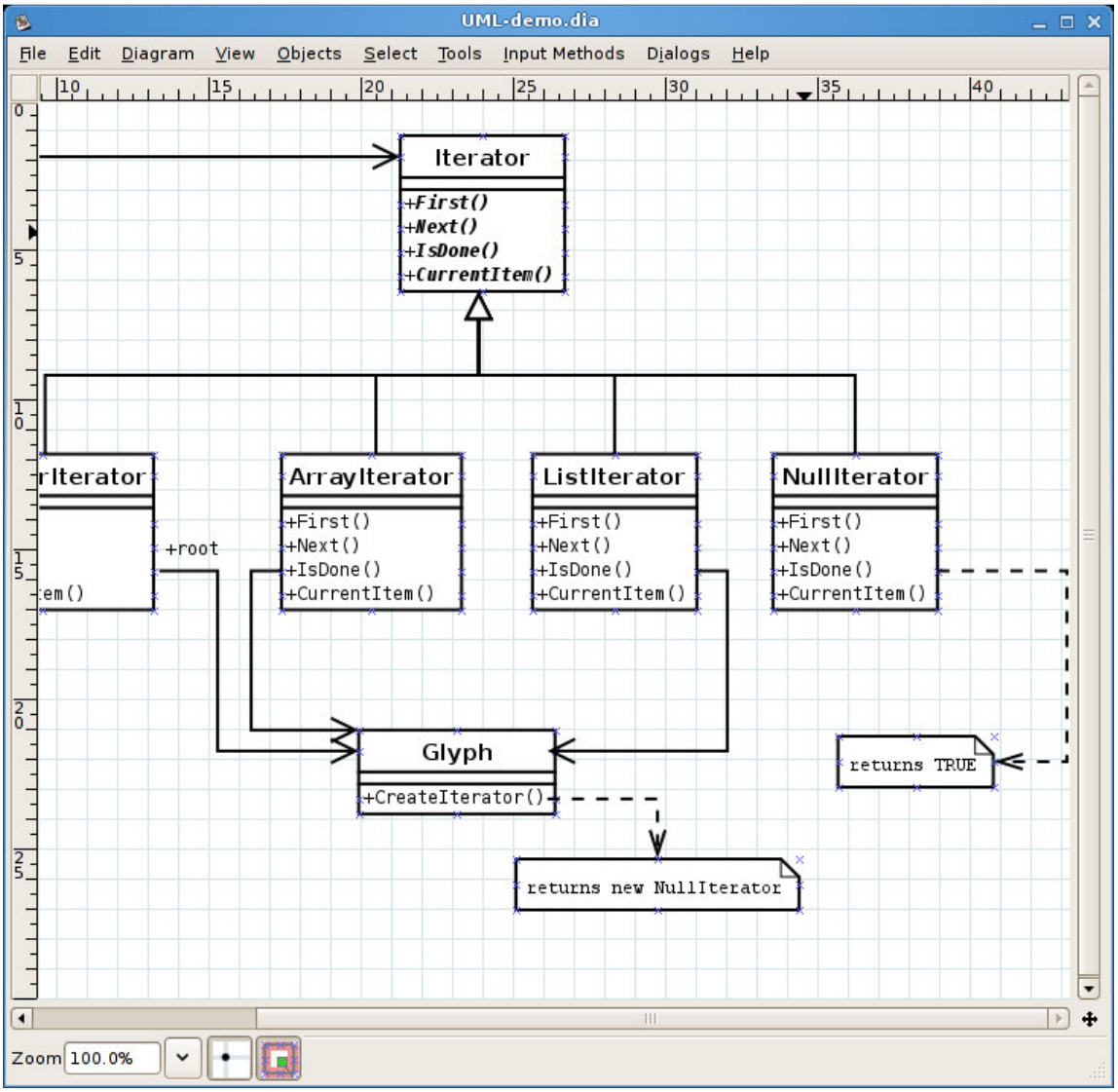
Lokalizace do češtiny: Ano, volitelná.

Platforma: Linux, Unix a Windows

Podpora diagramů: UML diagramy, E-R diagramy, vývojové diagramy, modely procesů, časový diagram (chronogram), diagramy spolupráce, diagramy tříd, stavový diagram, etc.

[144] http://www.altova.com/matrix_u.html,
http://www.altova.com/products/umodel/uml_tool.html

Obr. 32: Dia[145]



Z rodiny malých nástrojů jsem vybrala tento nástroj, který může sloužit jako varianta pro malé firmy, nebo firmy, které teprve začínají koketovat s možností CASE. Nástroj není vzhledem ke své jednoduchosti předpokládán k nasazení ve velkých firmách, stále ale může nabídnout mnoho zajímavého a pokrýt tak méně náročné požadavky s výhodou nízkých pořizovacích nákladů.

Nástroj nerespektuje žádnou speciální metodiku, spíše implementuje široce uznávaný objektový model. Silnou stránkou tohoto nástroje je podpora překvapivě velkého množství modelů, které nejsou omezeny restriktivně prvky, které lze v jednotlivých modelech použít, ale jsou vytvářeny pomocí volně vložitelných objektů z knihoven objektů. Proto je možné vytvářet relativně velké

[145] http://live.gnome.org/Dia/Screenshots

množství rozličných modelů čistě použitím odpovídajících objektů a vazeb mezi nimi.

Nástroj postrádá možnosti svých robustnějších a vybavenějších kolegů, jako jsou překlad do export do zdrojových kódů programovacího jazyka, transformace jednotlivých modelů, nebo například reverse engineering, jehož podporu lze u profesionálních nástrojů dohledat. Nástroj je použitelný pouze k jednouživatelskému užití, centrální úložiště modelů do repository není podporováno, kooperaci je tedy nutné řešit nástroji třetích stran. Stejně tak nejsou implementovaná komunikační rozhraní pro napojení na jiný firemní software.

Masovější nasazení do plného provozu bych spíše doporučila zvážit, minimálně z důvodu mizivé business podpory ze strany tvůrce. Jeho nasazení je snad odůvodnitelné v případě, kdy firma očekává pouze grafické vyjádření výstupu bez podpory širší analýzy, v minoritních projektech nebo v malé firmě. Pro střední firmu je nástroj snad použitelný jako podklad, kdy na základě efektivnějších výsledků práce může dojít k argumentaci o zakoupení rozvinutějšího nástroje.I tak si myslím, že nástroj si své uživatele v cílové skupině najde, zejména pak dle mého názoru u jednotlivců, než celých týmů.

12.8. JUDE

Aktuální verze: JUDE Professional 5.2.2

Výrobce: Change Vision, Inc.

Distributor: pouze stažením ze stránek výrobce

Cena licence: za 1 licenci, dále množstevní sleva

JUDE Community – zdarma

JUDE Professional – $280 za časově neomezenou licenci, $120/$70/$40 za licenci na 12/6/3 měsíce

Demoverze: Ano, JUDE Professional 20 dní

Lokalizace do češtiny: Ne přímo do výrobce.

Platforma: Windows

Podpora diagramů: Diagram tříd, UseCase Diagram, Sekvenční diagram, komunikační diagram, Statechart Diagram, Diagram aktivit, Diagram komponent, diagram nasazení, Composite Structure Diagram*, Objektový diagram, Package Diagram, Eriksson-Penker Process Diagram*, Mind Map*,ER Diagram*, Flowchart*, DFD*

* pouze ve verzi Professional

Tento nástroj je relativní novinkou na trhu s CASE nástroji – firma Change Vision, která jej vyvíjí, byla založena teprve v roce 2006. Nástroj existuje ve dvou variantách – Community, která je distribuovaná zdarma na základě registrace na stránkách výrobce, a Professional s podporou většího počtu diagramů a s podporou dalších schopností programu.

Současná verze nástroje již částečně podporuje novinky uvedené ve standardu UML 2.0. Rády bych vyzdvihla možnost použití Eriksson-Penkerova procesního diagramu, což v této cenové třídě není příliš obvyklé.

Velmi zajímavá funkce je výstup z diagramů do produktů MS Office – Wordu a Excelu, souběžně se „standardními" výstupy do dokumentů formátu rtf a HTML. Nástroj je schopen diagramy exportovat jako obrázky (v případě exportu do HTML je zakomponuje do exportu).

Nástroj je schopen konvertovat jednotlivé diagramy mezi sebou.

Již stávající schopnosti nástroje nabízí pokrytí základních i pokročilých uživatelských potřeb. Lze se domnívat, že při zachování stoupající kvality a implementace dalších diagramů běžně používaných v jiných obdobných nástrojích, bude tento nástroj již brzy konkurovat nástrojům od firem, které na trhu CASE nástrojů operují podstatně delší dobu.

12.9. Sparx Enterprise Architect

Aktuální verze: Sparx Enterprise Architect 7.1

Výrobce: Sparx Systems

Distributor:

Cena licence: za 1 licenci, od pěti licencí je množstevní sleva

Corporate Edition Floating License (bez omezení na uživatele) – $335

Corporate Edition – $239

Professional Edition – $199

Desktop Edition – $135

Demoverze: Ano, 30 dní

Lokalizace do češtiny: Ne.

Platforma: Windows, Linux

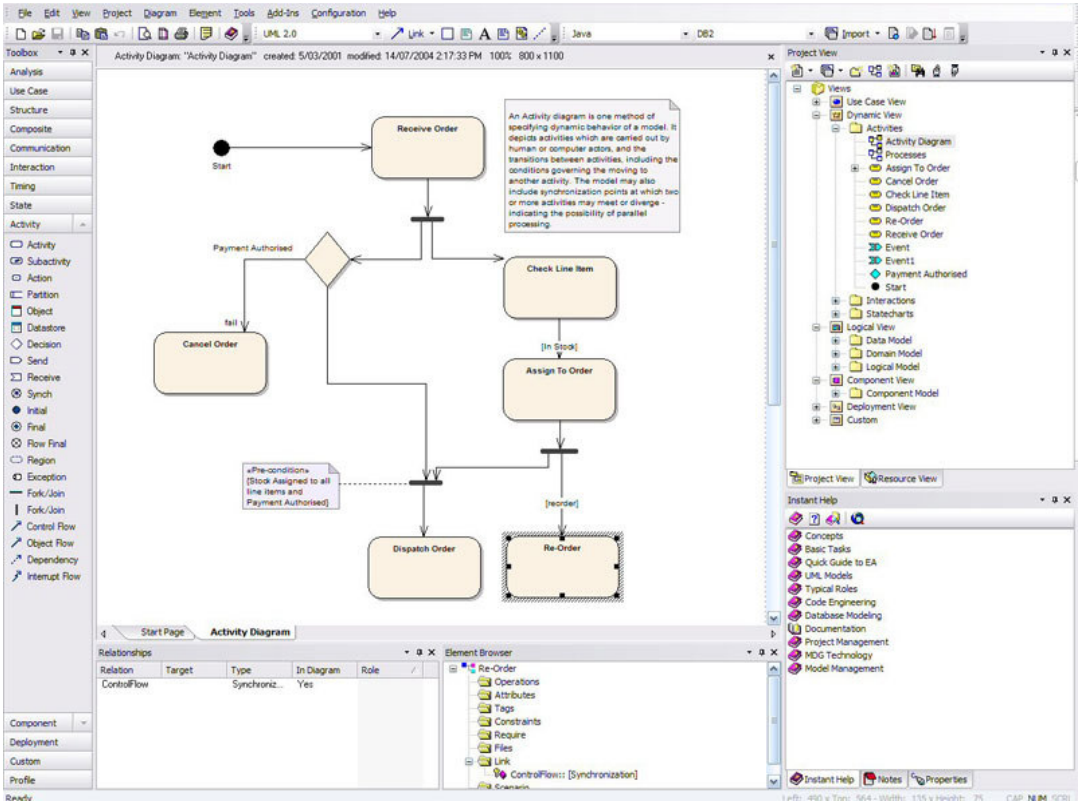
Podpora diagramů: Use Case Diagram, Sekvenční diagram, časový diagram (chronogram), diagram aktivit, diagram tříd (struktur), Package diagram, Objektový diagram, State Machine Diagram, diagram komponent, diagram nasazení, komunikační diagram, Interaction Overview diagram, Composite Structure

Stávající verze nástroje Sparx Enterprise Architect vychází primárně ze standardu UML 2.1, kde podporuje všech 13 jím definovaných diagramů (viz výčet výše). Současně výslovně podporuje metodiku MDA, na jejímž základě je schopen transformace modelů mezi sebou, výstupů do jazyků jako je DDL, Java, C#, EJB, XSD.

Nástroj je také schopen výstupu do programovacích jazyků a také reverzní engineering pro jazyky C++, Java, C#, VB.Net, Visual Basic, Pascal (Delphi), PHP, Python a ActionScript, přičemž lze volitelnými úpravami přidávat další podporované jazyky. Mimo jiné podporuje komunikaci s CORBA.

Tento nástroj je schopný integrace či užší spolupráce s nástroji jako je Eclipse, Microsoft Visual Studio 2005, Team Foundation Server, s jejichž pomocí dokáže spolupracovat při vývoji.

Obr. 33: Sparx Enterprise Architect[146]



Relativně silnou výhodou tohoto nástroje je schopnost generovat plně RTF dokumentaci z modelované problematiky, včetně testů, zdrojů, sledování změn apod.

Podpora SŘBD je relativně široká, samozřejmě přes ODBC, kde u korporátní edice lze přímo pracovat s databázemi jako MS SQL Server 2000 a 2005, MySQL, Oracle 9i a 10g, PosgreSQL, MSDE, Sybase Adaptive Server Anywhere, MS Access, Progress OpenEdge.

Tento nástroj vyniká svým přehledným uživatelským rozhraním, které je možné jednoduše upravovat a přizpůsobovat svým potřebám. Běžné jsou funkce jako dokování apod.

Díky příjemné ceně, podpoře forward i reverse engineeringu u velkého množství jazyků a podpoře širokého množství databází, stává se z Enterprise Architecta opravdu univerzální široce použitelný prakticky čistě vývojářský nástroj (což může být omezující), vhodný do libovolné střední firmy. V případě

[146] Zdroj obrázku:
http://www.sparxsystems.com.au/images/ea_screenshots/screenshot_diagram.jpg

koupě verze s podporou repository lze vcelku úspěšně uvažovat do jeho nasazení i do větších firem či projektů.

12.10. Vyhodnocení

Můžeme říci, že dnes je již trh dostatečně saturován ze strany dominantních firem s dostatečně silnými produkty. Produkty, které se svou cenovou politikou zaměřují na zákazníky ze segmentu velkých organizací, jsou v drtivě většině případů svázány s dalšími nabízenými nástroji v rámci několika komplexních balíčků. Díky těmto funkčním celkům jsou firmám nabízena konceptuální řešení problematiky nejen v úseku modelování systémů, ale navíc v celých vedlejších oblastech. Obecně se dá říci, že u velkých hráčů na trhu došlo k výrazné orientaci na přidanou hodnotu ke svým produktům, která je pozorovatelná v několika směrech – rozšiřitelnost nástroje, uživatelská přívětivost, nadstandardní funkce (za tyto si ale často bude muset budoucí uživatel připlatit).

V segmentu menších nástrojů se stabilně drží několik firem, které nabízejí řešení orientující se na zákazníka, s vybroušenou uživatelskou podporou a zvládnutou agendou, opírající se především o UML diagramy. Vlastní metodiky si nástroje pro střední segment trhu nemohou dovolit kvůli zvýšeným nákladům na školení zaměstnanců.

Do nástrojů pro jednotlivce se vešly ty výrobky, které dostačují pro jednoduché použití, nebo vynikají cenou. Již vzpomínaný produkt Dia je příkladem, že existují i produkty pro oblast trhu kopírující SOHO s produkty určenými nenáročným uživatelům.

Nemyslím si, že by do trhu s CASE nástroji v nejbližší době výrazně promluvila nová firma se svým produktem. V daných kategoriích je již celkem plno, CASE nástroje se postupně integrují do firemních IT struktur; nové možnosti vývoje závisí prakticky jen na nových metodikách, nebo se jedná o zlepšení kooperace a komunikace mezi prvky dosavadního systému – není tedy prozatím nic, čím získat stávající uživatele konkurenčních firem, ani nic, na co nalákat potenciální nové zákazníky. Zvýšení komplexity nástrojů ve středním segmentu by bylo kontraproduktivní z důvodu faktického opuštění tohoto segmentu a vzhledem k nízké migraci mezi nástroji ve větších firmách je zisk nového trhu nepravděpodobný.

Následující srovnávací tabulka vynáší podstatné vlastnosti výše popisovaných CASE nástrojů. Z důvodu přehlednosti jsem použila sloučení podporovaných diagramů do skupin dle jejich použití a doplnila dalšími informacemi.

Tabulka byla převzata a rozšířena o nová zjištění. [147]

[147] Čáp, J. Obrázek, M. Růžek, P. Turek, J. Smetana, J. Přehled nástrojů CASE na tuzemském trhu [online]. VŠE 2008. Dostupné na: http://panrepa.org/CASE/jaro2008/case_jaro2008.pdf

Tab. 1: Porovnání jednotlivých nástrojů

	Altova UModel 2008	DIA	IBM Rational ROSE	Power Designer	MS Visio	Select Architect	Oracle Designer	Sybase Power Designer	Sparx Enterprise Architect	JUDE
Umožňuje modelovat strukturu organizace	✔	✔	✔	✔	✔	✔	✔	✔	✔	
Podporuje modelování okolí	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Hrubý návrh IS (architektura, SW, HW)	✔		✔	✔	✔	✔	✔	✔	✔	✔
Detailní návrh – metodiky (UML)	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Podpora metamodelování	✔		✔	✔			✔	✔	✔	
Evidence funkcí a procesů v podniku	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Podporované SŘBD	✔		✔	✔	✔	✔	✔	✔	✔	✔
Monitoring – zasílání informací			✔			✔	✔		✔	
Reporty (statistiky, info o provozu)	✔		✔	✔	*	✔	✔	✔		✔
Generování manuálu pro uživatele	✔		✔	✔	*	✔	✔		✔	
Generování kostry kódu	✔		✔	✔	✔	✔	✔	✔	✔	
Bezpečnost – přístupová práva	✔		✔	✔	✔	✔	✔	✔	✔	✔
Verzování	*		✔	✔	*	✔	✔	✔	✔	

	Altova UModel 2008	DIA	IBM Rational ROSE	Power Designer	MS Visio	Select Architect	Oracle Designer	Sybase Power Designer	Sparx Enterprise Architect	JUDE
Podpora práce více uživatelů	✔		✔	✔	✔	✔	✔	✔	✔	✔

*) na rozdíl od informací poskytovaných zdrojem case_jaro_2008.pdf se mi při analýza nástroje nepodařilo tuto informaci potvrdit

Zdroji čerpání informací byly především stránky výrobců CASE nástrojů a dále webové stránky věnující se problematice metodik a nástrojů pro vývoj informačního softwaru[148].

[148] http://www-128.ibm.com/developerworks/rational/library/05/329_kunal/, http://www-306.ibm.com/software/awdtools/modeler/swmodeler/features/index.html, http://www3.software.ibm.com/ibmdl/pub/software/rational/web/datasheets/rsm.pdf, http://www.lbms.cz/Nastroje/index.html, http://objekty.vse.cz/Objekty/MetodikyANotace-NastrojeSrovnani, http://www.prikryl.cz/cze/htmlseminarka.php?id=case, http://panrepa.org/CASE/jaro2008/case_jaro2008.pdf

13. Metodika pro výběr vhodného CASE nástroje

13.1. Úvod

V České republice existuje Směrnice pro hodnocení a výběr CASE nástrojů, jmenovitě ČSN ISO/IEC 14102. Tato česká norma přebírá anglickou verzi mezinárodní normy ISO/IEC 14102:1995, která má český status technické normy. Informace o této normě naleznete v příloze v podkapitole 15.3. Vzhledem k tomu, že tato směrnice pochází z roku 1995, nesplňuje již požadavky dnešní doby.

Díky posunu požadavků a dovedností očekávaných od CASE nástrojů v průběhu uplynulých deseti let od vydání této specifikace se stala tato norma poněkud zastaralou. Zejména fakt, že její koncept, který je z důvodu prevence oproti zastarání vzat velice obecně, obsahuje dnes již poněkud archaické prvky, resp. předpokládá rozšíření schopností CASE nástrojů do oblastí, které jsou dnes samozřejmostí nebo jsou stále nedosažitelné, posunuje hodnotící kritéria mimo dostatečně přesnou oblast dnešního zájmu a situaci na dnešním trhu nástrojů.

Zhodnocení nedostatků této normy dala vzniknout mému návrhu metodiky, která vede čtenáře jasnými příklady hlavně v oblasti hodnotících kritérií a zejména obsahuje jednoduchý dvoustupňový hodnotící algoritmus, zatímco norma v příloze B pouze odkazuje na několik doporučených hodnotících postupů. Z důvodu návaznosti na různé hodnotící metody neobsahuje metodu pro získání dostatečného množství relevantních dat pro rozhodování.

Metodika popisuje kroky, kterými je potřeba postupovat při výběru nejvhodnějšího nástroje pro unikátní potřeby firmy, včetně definic požadavků a jejich vyhodnocení, tento problém řeší a popisuje jednotlivé relativně jednoduché potřebné úkony pro výběr požadovaného nástroje.

V mé metodice, při srovnání s výše jmenovanou normou, kladu větší důraz na prvky, se kterými se běžný manažer dostane do kontaktu, jako je myšlenka na outsourcing tohoto výběru CASE nástroje a kooperace se zaměstnanci během celého procesu výběru. Dle mého názoru je má metodika komplexnější v těchto oblastech a to při zachování dostatečné jednoduchosti. Také proto bych řekla, že narozdíl od normové metodiky je má metodika uživatelsky příjemnější.

Srovnám-li mnou vytvořenou metodiku s metodikami pro vedení projektů, přikláním se spíše k cestě méně restriktivní, která předkládá uživateli spíše doporučení při volbě metod pro splnění jednotlivých etap popisovaných touto metodikou. Přísně restriktivní forma metodiky také ani není možná z důvodu jejího velmi širokého okruhu uplatnění v různých firmách s širokým spektrem jednotlivých počátečních situací u projektů zavádění do firem a pro firmy

a potřeb ve firmě, která se pro CASE nástroj rozhoduje. Proto se jedná spíše o soubor metod a postupů odvozených z praxe, best practices získatelných z dostupných zdrojů, a jiných metodik zabývajících se podobným tématem.

Na příkladech v předcházející kapitole je zřetelně vidět, že na trhu existuje nepřeberné množství jednotlivých CASE, které splňují požadavky budoucích zákazníků. Vzhledem k tak velkému množství nástrojů na trhu a vzhledem k jejich unikátnosti je relativně obtížné nalézt nejvhodnější nástroj pro danou firmu. Jednoduché objektivní porovnání se tak stává problémem, snaha kategorizovat tyto nástroje naráží na fakt, že některé nástroje se k této hranici blíží, případně ji překračují.

Metodika popisující kroky, kterými je potřeba postupovat při výběru nejvhodnějšího nástroje pro unikátní potřeby firmy, včetně definic požadavků a jejich vyhodnocení, tento problém řeší a popisuje jednotlivé relativně jednoduché potřebné úkony pro výběr požadovaného nástroje.

Srovnám-li mnou vytvořenou metodiku s metodikami pro vedení projektů, přikláním se spíše k cestě méně restriktivní, která předkládá uživateli spíše doporučení při volbě metod pro splnění jednotlivých etap popisovaných touto metodikou. Přísně restriktivní forma metodiky také ani není možná z důvodu jejího velmi širokého okruhu uplatnění v různých firmách s širokým spektrem jednotlivých počátečních situací u projektů zavádění do firem a pro firmy a potřeb ve firmě, která se pro CASE nástroj rozhoduje. Proto se jedná spíše o soubor metod a postupů odvozených z praxe, best practices získatelných z dostupných zdrojů, a jiných metodik zabývajících se podobným tématem.

Jednotlivé operace výběru vhodného CASE nástroje do firmy jsou sloučeny do logických celků – etap, jejichž vzájemná propojení jsou uvedena v následujícím schématu. Následující obrázek nám znázorňuje doporučený postup v jednotlivých etapách výběru CASE nástroje, které budou podrobněji rozebrány v následujících podkapitolách. Jednotlivé etapy a jejich vzájemná souslednost tohoto modelu je pouze doporučení, ke kterému by měl manažer, rozhodující o nasazení CASE systému a zodpovědný za jeho výběr, větší měrou přihlížet, ale ve specifických situacích (jako například u tabulky hodnotících kritérií) se předpokládá použití vlastní logiky pro pokrytí unikátních potřeb podniku. Manažer si také musí nastavit jednotlivé kvalitativní ukazatele použité v etapách výběru, případně analogicky libovolný potřebný kvalitativní ukazatel přidat.

Obr. 34: Etapy pro výběr vhodného CASE nástroje

Zvláštní případ nastává, když je výběru a nasazení systému CASE outsourcován na jinou firmu. Tento případ bude rozebrán na konci této kapitoly s výčtem změn a upozorněními k tomuto způsobu řešení.

Na možnost, kdy firma před zahájením projektu již CASE nástroj vlastní, je během práce upozorněno jmenovitě u jednotlivých etap, které se touto situací zabývají.

13.2. Přehled metodiky

V první řadě je potřeba, aby si manažer určil, zda a jak robustní CASE nástroj potřebuje. Pro vývoj malých aplikací lze předpokládat, že pořízení komplexního modelovacího nástroje je spíše zbytečnou investicí. Reálné zhodnocení situace přináší reálná očekávání na CASE nástroj a reálné nároky na jeho vlastnosti. Malá firma vyvíjející jednoduchý software bude zřejmě inklinovat k použití jednoduššího nástroje, jelikož pokročilé funkce nevyužije.

První etapou výběru CASE nástroje je sběr interních informací. V jeho průběhu začíná manažer získávat informace potřebné k stanovení požadavků na pořizovaný CASE nástroj. Mimo jiné je důležité zjistit informace o ostatních používaných softwarových nástrojích z důvodu jejich propojení s pořizovaným CASE nástrojem, dále je důležité určit, podle jaké metodiky provádí firma modelování, a podobně.

Získané informace nám pomohou v další etapě, kde určujeme, jaké vlastnosti očekáváme od pořizovaného nástroje. Rozsah požadavků je relativně široký, od požadavku na podporované metodiky, standardy, se jedná o to, v jaké fázi vývoje očekáváme využívání CASE nástroje, jakým způsobem a s jakým softwarem má být CASE nástroj schopen komunikovat, až po hodnocení záruční a pozáruční podpory. V popisu etapy jsou tyto vlastnosti rozděleny do čtyř oborů a z těchto požadavků je stanovena hodnotící tabulka. Tuto tabulku chápu jako jádro své práce, díky jednoduchému vyplnění požadované funkcionality z nabízených položek, které dle mého názoru pokrývají většinu požadavků, které lze na vybíraný CASE nástroj mít. Současně určujeme důležitost jednotlivých kritérií a jejich požadovanou kvalitu.

Hodnotící tabulka je nám vodítkem pro sběr dat a provádění hodnocení CASE nástrojů. Informace získáváme bez přímého kontaktu s CASE nástrojem. Vzhledem k množství nástrojů zde nelze řešit subjektivní vlastnosti nástroje, jako je jeho ovládání, apod. – tuto důležitou složku kvality práce s CASE nástrojem nechávám až na užší výběr.

V širším vyhodnocení dostáváme z relativně rozsáhlé množiny CASE nástrojů cca pět nejlepších. Metodou výběru je eliminace nevyhovujících nástrojů a výběr pěti nejlepších nástrojů srovnaných podle celkového hodnocení.

Naopak oproti přemíře nástrojů v předchozí etapě je v užším výběru díky předvýběru pouze několik nástrojů, které nyní již můžeme testovat osobně. V této etapě se hodnotí zejména to, jakým způsobem vyhovuje daný CASE nástroj našim potřebám pomocí metody přímého experimentu. Nyní již hodnotíme praktickou použitelnost jednotlivých nástrojů ve firmě, řešíme jaká verze nástroje uspokojuje naše požadavky a provádíme srovnání pořizovacích nákladů na pořizovaný CASE nástroj. Na základě těchto poznatků docházíme k výběru požadované verze CASE nástroje.

Nyní tedy k jednotlivým etapám navrhované metodiky podrobně.

13.3. Zjištění potřeby nového CASE nástroje

Tato etapa výběru CASE nástroje usnadní manažerovi rozhodnutí, zda firma potřebuje CASE nástroj a nakolik je jeho nasazení přínosem, dále také k čemu bude tento CASE nástroj použit.

13.3.1. Podnik nepoužívá žádný CASE nástroj

Nyní se pokusíme zodpovědět otázku, za jakých podmínek dochází ke vzniku požadavku na nový CASE nástroj, resp. jak manažer dojde ke zjištění, že jeho podnik CASE nástroj opravdu potřebuje. Předpoklady, které manažerovi signalizují, že je třeba doplnit portfolio programového vybavení o CASE nástroj, je možné, abychom obecně kvalifikovali kombinací několika možných ukazatelů:

- Pokud se podnik zabývá ve svém core procesu, či v jeho části, modelováním (pravděpodobně za účelem vývoje specializovaného software).
- Pokud se podnik identifikuje s následujícími kritérii, pak potřeba CASE nástroje stoupá spolu s mírou identifikace (v závorce uvádím možný problém, který může nastat při nenasazení CASE nástroje):
    1. Podnik vyvíjí různé zakázky pro různé zákazníky, čímž narůstá množství již splněných zakázek (může tak zbytečně docházet ke ztrátě informací o jednotlivých projektech a ke ztrátě znalostí z jejich řešení).
    2. Podnik má větší počet zaměstnanců, kteří provádějí analýzu a vývoj, nebo plánuje nárůst jejich počtu (může být problém s předáváním znalostí v řešení problematiky a problém s nekonzistencí mezi zaměstnanci v rámci projektů).
    3. V podniku je pouze jeden nebo velmi úzký okruh zaměstnanců, kteří provádí analýzu a vývoj (může dojít k velkým problémům, pokud zaměstnanec odejde nebo zůstane dlouhodobě nemocný, protože jen daný zaměstnanec své práci rozumí a je těžké jej někým nahradit).
    4. Produkt vyvíjený firmou nabývá na komplexnosti a složitosti (může dojít k nekonzistenci mezi jednotlivými částmi produktu).

Za předpokladu, že je manažer schopen identifikovat určitou situaci popsanou výše, nebo je schopen identifikovat hrozící rizika, která plynou z nenasazení CASE nástroje, pak je vhodné, aby zvážil nasazení CASE nástroje.

CASE nástroj není nutné nasazovat na všechny počítače všem zaměstnancům, což se týká nejen velkých firem, které mají pouze oddělení vyvíjející aplikace pro interní použití, ale také pro firmy, které se na vývoj aplikací specializují. Jelikož je manažer zodpovědný za rozhodnutí, jaký CASE nástroj vybere, má za úkol alespoň rámcově odhadnout, na kolika počítačích se bude CASE nástroj využívat a kolik uživatelů jej bude aktivně používat.

Je nutné si uvědomit, že motivace pro zavedení CASE nástroje do firmy a forma jeho použití může být v každém podniku různá a podle toho stanovovat i podmínky pro budoucí požadovaný CASE nástroj. Sama bych

stanovila plánované použití CASE nástroje v několika úrovních postupně podle úrovně využití nástroje (vycházím ze studie Fuggetta[149]):

- CASE nástroj bude použit jako prostředek pro grafické ztvárnění informací podporujících komunikaci mezi dvěma subjekty – tato možnost neočekává od nástroje výraznou komplexnost, kontrolu diagramů, jejich vzájemnou konverzi ani podporu sdílení diagramů mezi více uživateli. Pokrytí požadavku může zprostředkovat jakýkoliv freeware nástroj, dokonce lze tvrdit, že by na splnění požadavků kladených na CASE nástroj stačil prakticky libovolný grafický editor s podporou manipulace s objekty, Tento signál by ale manažer měl brát jako podnět k zamyšlení nad potřebou nasadit do procesu modelování nástroj sofistikovanější, který by se kromě procesů, pro něž jsou primárně CASE nástroje určeny, dal využít i k výše popisované činnosti.
- CASE nástroj je plánován pro použití jako prezentační nástroj mezi analytiky a vývojáři, případně jako úložiště znalostí ze současných projektů – v tomto případě je vysoce pravděpodobné, že již k modelování dochází a to velice často svépomocí na papír nebo jiné obdobné médium, tedy CASE je potřeba pro zachycení modelů projektů pro jejich další použití. Obecně lze tuto potřebu pokrýt CASE nástrojem, který zvládá modelovat jednotlivé modely, pravděpodobně i dokáže tyto modely umístit do centrálního úložiště, aby k nim měli přístup všichni oprávnění zaměstnanci. Ve většině případů bude stačit základní verze libovolného komerčního CASE nástroje.
- CASE nástroj bude použit jako komplexní modelovací nástroj – úplné využití špičkového CASE nástroje zahrnuje modelování jednotlivých stupňů abstrakce reality, automatické kontroly konzistence modelů a jejich vzájemné navázání, podpora meta-modelování a CABE přístupů. Tato kombinace bude volena týmy, které od CASE nástroje očekávají vyšší odolnost vůči chybám během analýzy a berou jej jako plnohodnotný nástroj použitelný ve všech fázích vývoje projektu.

Při rozhodování o tom, zda nasadit CASE nástroj či ne, můžeme vycházet z principu zvyšování efektivity práce,[150,151] případně z výše uvedených benefitů viz podkapitola 9.7.1. Pro zamyšlení nad potřebou nasazení CASE nástroje je možná lépe, než klasifikovat přínosy zavedení CASE nástroje, také definovat ztráty, které s sebou nese jeho nenasazení. Manažer by měl spíše zvážit, zda může nenasazením dojít k možnosti ztráty konkurenceschopnosti na trhu, snížení flexibility společnosti, apod.

---

[149] Fuggetta, A. a Classification of CASE technology. Computer, December 1993, str. 25 - 38

[150] Banker, R.D. Kaufmann, R.J. Reuse and Productivity in Integrated Computer-aided Software Engineering: An Emperical Study, MIS Quaterly, 15:3, 1991, str. 375 - 401

[151] Finlay, P.N. Mitchell, A.C. Perceptions of the Benefits from the Introduction of CASE: An Emperical Study, MIS Quaterlym 18:4, 1994, str. 353 - 370

13.3.2. Podnik již určitý CASE nástroj používá

Na rozdíl od již vyřešené problematiky nově nasazovaného CASE nástroje, je situace v momentě, kdy firma již používá stávající CASE nástroj, který jí nevyhovuje, daleko komplexnější.

V případě, kdy již ve firmě je CASE nástroj přítomen, je nutné najít odpověď na otázku, jakou přidanou hodnotu nám nový CASE nástroj přinese, zda je nevyhnutelné pořizovat nový nástroj od jiného výrobce a zda tedy není výhodnější provést úpravy stávajícího nástroje, např. jeho upgrade, nebo dokoupit případně vyšší licenci, rozšiřující moduly a podobně.

Další okruh otázek se týká problematiky, proč chceme opustit stávající řešení. Například může nastat problém, že je nevyužívané, z důvodu, že nevyhovuje novým nárokům, které se vyskytly během jeho používání, že je celkově zastaralé a již neexistuje výrobce, který nadále tento software podporuje.

Další možnost je zastarání CASE nástroje oproti jiným softwarovým komponentám používaným v podniku, díky kterému například přestane fungovat funkční propojení na nové nástroje, případně vypršení licence. Mezi jinými je nutné zvážit, zda není problém v kroku odepsání nástroje z majetku společnosti, případně zda se licence k jeho použití neváže na nějaký jiný stávající majetek, který by bylo nutné vyřadit také, což by vedlo ke zvýšení nákladů na pořízení nového majetku.

Primárně je nutné určit důvod neúspěchu stávajícího CASE nástroje a snažit se odstranit jeho příčinu. Metoda odstranění příčiny neúspěchu pak může být porovnána s metodou koupě nového systému nezávisle na stávajícím a dle porovnání kladů a záporů lze dojít k výsledku, že výhodnějším se bude jevit právě řešení spočívající v odstranění příčin, díky kterým současný nástroj není dostatečně či efektivně využíván.

V následující tabulce uvádím několik možných problémů, které se mohou vyskytnout u stávajícího modelovacího nástroje a možná řešení bez nutnosti pořízení nového CASE nástroje.

Tab. 2: Možné důvody neúspěchu při používání aktuálního modelovacího nástroje a možná řešení bez nutnosti pořízení nového modelovacího nástroje

Definice problému	*Možný způsob vzniku problému*	*Možná identifikace problému*	*Možnosti řešení problému*
Nástroj nepokrývá aktuální potřeby firmy	Firma změnila zaměření, Přibyly požadavky na nové typy modelů	Zaměstnanci nejsou schopní vytvořit kvalitní modely	Upgrade / update stávajícího nástroje, Zakoupení nového CASE

Definice problému	*Možný způsob vzniku problému*	*Možná identifikace problému*	*Možnosti řešení problému*
Nástroj pozbyl možnosti efektivní komunikace s jinými nástroji	Zastarání CASE nástroje, Nové verze ostatních nástrojů nepodporujících stará komunikační rozhraní, Partneři pořídili nástroje s nekompatibilními výstupy	Ztráta možnosti komunikace mezi aplikacemi, Obtížnější komunikace s partnery, Chyby generované provozem aplikace	Identifikace problému v komunikaci, Úprava komunikačního rozhraní např. přidáním proprietálního protokolu (pokud lze) Zpětné začlenění do podnikové komunikační struktury
Neznalost programu, jeho ovládání, možností, neschopnost plného využití jeho potenciálu	Obměnou zaměstnanců na jakékoli z pozic, kteří nástroj používají, Špatné nastavení nástroje při/po koupi	Ověřením znalostí uživatelů, výstupů jejich práce, schopnosti rozumět modelovaným údajům Degradace CASE nástroje na kreslící nástroj	Proškolení stávajících zaměstnanců, zavedení školení pro nové zaměstnance pracující se SW, překonfigurování SW

Ráda bych k této tabulce upozornila na problém, který se týká neznalosti programu. Je to typický případ, na kterém lze demonstrovat nutnost analýzy správného důvodu selhání využití CASE nástroje při návrhu systémů, resp. odhalení pravdy v předkládaných informacích (předpokládám možnou situaci, že zaměstnanec raději označí nástroj za nevhodný k současné činnosti, než aby přiznal, že s jeho ovládáním není obeznámen. Odhalení správného problému pak může až několikanásobně snížit náklady na jeho odstranění.

Pokud se analýzou problému prokáže, že lze vyloučit problém na straně uživatele (např. výše zmiňovaná neznalost programu) a objektivně se tedy jedná o potřebu rozšíření funkcionality nástroje, lze tuto metodiku využít také. Stávající nástroj pak bude přidán jako jeden z kandidátů a bude porovnáván s ostatními produkty (viz. popis metodiky v etapách analýza trhu, širší a užší vyhodnocení).

Za předpokladu, že manažer na základě zjištěných informací dojde k závěru, že je žádoucí přemýšlet o nasazení nového CASE nástroje, je zejména nutné, aby byl vypracován projektový plán, či jiné předběžné zhodnocení, ve kterém shrne přínosy a rizika pro firmu vyplývající z nasazení či nenasazení nového CASE nástroje a další úkony potřebné k tomu, aby došlo k odsouhlasení projektu a jeho přidání do projektového portfolia firmy.

Veškerá argumentace, která byla použita při zdůvodnění o přidání projektu do projektového portfolia, je velmi často relevantní také jako podklad pro další etapy výběru nového CASE nástroje.

Tato etapa je ukončená se zhodnocením zodpovědného manažera, zda je potřeba pořídit nový CASE nástroj, a v případě potřeby nasazení nového

CASE nástroje jsou nashromážděny údaje a důvody pro zavedení projektu do portfolia firemních projektů, je stanoven termín provedení tohoto projektu.

13.4. Sběr interních informací pro výběr CASE nástroje

Tato etapa se zabývá stanovením požadavků na nově pořizovaný CASE nástroj. Informace zjišťované v této etapě lze rozdělit do tří okruhů, z čehož každý okruh může být zpracováván nezávisle na sobě, paralelně různými zaměstnanci či skupinou zaměstnanců. Prvním okruhem jsou:

1) informace o současném hardwarovém a softwarovém vybavení firmy. Tyto informace pokrývají technické požadavky na CASE nástroj. Je nutné zmapovat zejména následující údaje:
    2. na kolika počítačích bude nástroj používán,
    3. jaké je hardwarové a softwarové vybavení pracovní stanice,
    4. jaké další programy jsou ve firmě využívány, z pohledu navázání budoucího CASE nástroje, stávající komunikační politiky firmy, rozhraní jiných nástrojů, atd.

Tyto podklady slouží jako základ pro stanovení požadavků na CASE nástroj v příští etapě. Předpokladem je, aby CASE nástroj podporoval současný stav hardwaru a softwaru ve společnosti Další okruhy jsou:

2) Informace o budoucích uživatelích CASE nástroje (případně stávajícího nevyhovujícího CASE nástroje), jako např.:
    1. počet uživatelů, kteří budou nástroj používat a na jaké úrovni bude jeho využití u jednotlivých uživatelů,
    2. definice potřeb a očekávání zaměstnanců vůči novému CASE nástroji,
    3. dosavadní zkušenosti uživatelů s CASE nástroji, jejich výtky a doporučení.
3) Informace vedoucí k definici objektivních požadavků na CASE nástroj, jako např.:
        1. výčet diagramů, na které jsou nyní používány alternativní metody vytváření
        2. podle jakých metodik firma pracuje

Tyto informace slouží nejen jako podklad pro hodnocení CASE nástroje, ale také pro budoucí odhad nákladů na pořízení CASE nástroje a pro zajištění zpětné vazby od uživatelů směrem k managementu. Zjištění zaměstnanci s pozitivním přístupem k nasazovanému nástroji mohou být použiti jako propagátoři jeho nasazení, zkušební pracovníci, odhalující chyby, atd. Více o této problematice v popisu etapy kooperace pracovníků.

V této části by měl manažer provést úkony, které běžně provádí při každém projektu, tj. stanovit nákladů projektu, časové náročnosti etap projektu, přidělení rolí a zodpovědnosti pracovníkům v rámci projektu, atd. Jakmile toto

manažer provede, měl by mít již dostatečné informace potřebné k definování požadavků na CASE nástroj, spuštěný projekt výběru a nasazení CASE nástroje. Současně již definoval metody, které budou použity v rámci plnění následujících etap.

13.5. Definování požadavků na CASE nástroj

Tato etapa slouží k definici pravidel, podle kterých firma vybírá nejvhodnější CASE nástroj. Takto nastavené podmínky pro CASE nástroje pak budou uplatňovány v dalších etapách vývěru. Během sestavování hodnotící tabulky lze vycházet z požadavků, které byly upřesněny v etapách předchozích:

- argumenty použité během vzniku projektu jako jsou potřeby pro pořízení nového CASE nástroje
- technické parametry, které musí CASE nástroj splnit, a další informace, získané v předchozí etapě

V této etapě je nutné definovat požadavky na CASE nástroj. Požadavky by měly vyplynout z výše uvedených etap, komunikace s budoucími uživateli (více v popisu etapy Kooperace a komunikace se zaměstnanci) a objektivními požadavky na funkčnost nástroje. Takto získané požadavky na CASE nástroj lze rozdělit obecně do čtyř skupin:

- Vlastnosti nástroje – do této skupiny hodnotících kritérií patří veškerá kritéria, která se vztahují k požadované funkčnosti CASE nástroje z pohledu uživatele (na různých úrovních jeho použití), jeho rozšiřitelnost, apod.
    - Podpora diagramů – jako hlavní vlastnost CASE nástroje je podpora modelování jednotlivých diagramů a modelů, proto má tato skupina dominantní postavení
    - požadované schopnosti nástroje – konverze diagramů, podpora metamodelování, verzování, apod.
    - jaké metodiky a jaké notace nástroj podporuje, zda je schopen jejich použití kontrolovat, apod.
- Komunikace a rozhraní – tato skupina obsahuje požadované vlastnosti vzhledem k podpoře přenositelnosti dat se zákazníky, možnostem exportu dat do jiných programů a provázání CASE nástroje s již existujícími nástroji ve firemním prostředí. Dále se v této skupině eviduje, zda je CASE nástroj schopen (polo)automatického generování dokumentace, v jakém formátu je pro nás nejpřijatelnější apod.
- Podpora – tato skupina definuje přidané služby dodávané k nástroji, jako je instalace, školení uživatelů a všeobecně služby přidané k nástroji při jeho pořízení

- Technické požadavky – v této skupině se uvádějí vlastnosti, které je nutné splnit, nebo sledované z hlediska technického charakteru CASE nástroje, mezi jinými jde typicky o požadovaný software k provozu, operační systém, zálohování, apod. do této skupiny jsem také zařadila relativně důležité

Každý ukazatel (neboli hodnotící kritérium) je z hlediska potřeby firmy subjektivně hodnocen různou důležitostí, protože každá firma je unikátní. Abychom mohli tuto jedinečnost definovat, budeme používat váhu ukazatele, která může nabývat hodnot 0 - 10. Hodnota 0 značí, že tento ukazatel není absolutně pro výběr relevantní. Naopak hodnota 10 značí klíčovou vlastnost CASE nástroje.

Stanovení Vah ukazatele je zpětnou kontrolou, zda v hodnotící tabulce jsou relevantní hodnotící kritéria – v momentě, kdy se stane, že kritériu je přiřazena hodnota váhy 0, toto kritérium postrádá smysl a je možné jej z hodnocení vypustit.

Hodnota označovaná jako „Požadované minimum“ slouží v budoucím hodnocení jako filtr pro nevyhovující nástroje. Pokud tedy nástroj postrádá nějakou vlastnost, nebo ji nemá v požadované kvalitě, je možné jej během fáze výběru vyřadit nezávisle na tom, jaké výsledky vykazuje v jiných kategoriích. Hodnota „Požadované minimum“ vychází ze škály hodnocení, stejně jako ono může nabývat hodnot 0 - 10.

U každého ukazatele, případně skupiny či podskupiny, je nutné určit, zda má minimální hranici bodů, kterou musí daný nástroj splňovat. Vycházejme ze skutečnosti a reálných potřeb firmy. Například pokud firma pro svou práci potřebuje BPM[152], nastaví si jako potřebné minimum u ukazatele grafu BPM na hodnotu 1. V případě, kdy je požadována 100% funkčnost nějaké komponenty, je nutné rozpracovat jednotlivé skupiny do takových podrobností, aby bylo možné obsáhnout přímo sledovaný ukazatel, který se poté hodnotí.

Důvodem tohoto rozpadu ukazatelů na skupiny je požadavek, aby potřebné minimum bylo adresováno přímo na jednu určitou položku, ne na skupinu položek – pak by se mohlo stát, že skupina, ve které nebyl použit rozpad na nižší ukazatele, dosáhne vyššího hodnocení, než je nastavené potřebné minimum, a nástroj přitom postrádá námi deklarovanou bezprostředně požadovanou funkcionalitu. Další nutnost rozkladu ukazatele na skupinu je v případě, kdy v nově vytvořené skupině má jeden nebo více ukazatelů v této skupině výrazně odlišnou váhu než ostatní.

Příkladem může být např. hodnotící kritérium „Síťová správa.“ Pokud nedojde k jejímu rozpadu na ukazatele složky „Vzdálená instalace“ a „Vzdálená správa“, tak by se při zpracování mohlo pominout to, že nástroj například není schopen vzdálené bezobslužné instalace, což je například

[152] BPM je zkratka pro "Business Process Modeling" neboli modelování podnikových procesů. Jedná se o aktivitu, která popisuje současný stav podnikové entity a požadovaný budoucí stav.

požadavek firemní politiky a nástroj by byl ve světle ostatních vlastností spojených se síťovou správou ohodnocen vysoko, a přesto by nesplňoval požadovaná kritéria.

Vzhledem k tomu, že některá hodnotící kritéria nenabývají číselných hodnot na stupnici 0 - 10, ale slovního vyjádření stávajícího stavu, je pro další postup nutné tento stav nějak vyhodnotit. Pro to je potřeba k některým kritériím vytvořit tabulku s převodem mezi výčtem hodnot, které může kritérium nabývat, a kvalitativní měrnou hodnotou prezentovanou škálou 0 - 10. Příkladem může být kritérium „Operační systém", pro který může dle možností firmy být vhodná jedna nebo více možností se svojí relativní vhodností.

Bohužel díky tomu, že jde o unikátní vlastnost každého zpracování závislé na stávající situaci ve firmě a jejích požadavcích na pořizovaný CASE nástroj, nemohu nyní obecně uvést obecně platné tabulky pro převod mezi stávajícím stavem a výší hodnotícího kritéria.

Tato tabulka k hodnocení jednotlivých ukazatelů bude následně závazná pro pracovníky hodnotící jednotlivé CASE nástroje. Kromě čistě subjektivních ukazatelů lze tuto legendu vypracovat pro všechny hodnotící kritéria. Příkladem, jak lze rozpracovat hodnotící kritérium, je následující tabulka:

Tab. 3: Příklad rozpracování hodnotícího kritéria

Ukazatel	***Hodnota***	***Vyjádření***
serverový OS	0	nemá podporu centrálního úložiště dat
	2	OS Unix placený – nutno dokoupit HW
	3	OS Linux freeware – nutno dokoupit HW
	7	libovolný jiný OS Windows než používaný
	10	používaný OS Windows
Podporované metaCASE	0	metaCASE není implementováno
	10	metaCASE je implementováno

V následující tabulce je zobrazen doporučený tvar hodnotící tabulky s obecnými požadavky v jednotlivých skupinách, jak tato metodika doporučuje. Z požadované podpory modelů jsem použila výčet modelů ze standardu UML 2.0, který je možné samozřejmě podle potřeby rozšířit o další nedefinované v běžných metodikách, jako např. Eriksson-Penkerův procesní diagram, Mind map, atd.

Tab. 4: Doporučená podoba hodnotící tabulky

Ukazatele / Hodnotící kritéria	***Váha ukazatele***	***Požadované minimum***	***Hodnocení***

Ukazatele / Hodnotící kritéria	*Váha ukazatele*	*Požadované minimum*	*Hodnocení*
1. Vlastnosti nástroje			
1.1 Diagramy a modely			
1.1.1 Diagram tříd			
1.1.2 Composite Structure Diagram			
1.1.3 Diagram komponent			
1.1.4 Diagram nasazení			
1.1.5 Objektový diagram			
1.1.6 Package diagram			
1.1.7 Diagram aktivit			
1.1.8 Use Case Diagram			
1.1.9 State Machine Diagram			
1.1.10 Sequence Diagram			
1.1.11 Communication Diagram			
1.1.12 Intercation Overview Diagram			
1.1.13 Timing Diagram			
1.2 Podporované metodiky			
1.2.1 metodika RUP			
1.2.2 metodika EUP			
1.2.3 metodika PDIT			
1.2.4 metodika BSP			
1.3 Podporované standardy			
1.3.1 UML 2.0			
1.3.2 BPEL4WS			
1.3.3 BPMN			
1.3.4 DTD			

Ukazatele / Hodnotící kritéria	*Váha ukazatele*	*Požadované minimum*	*Hodnocení*
1.3.5 IDEF			
1.3.6 RDBMS			
1.4 Nástroj lze přiřadit do skupiny			
1.4.1 Pre CASE			
1.4.2 Upper CASE			
1.4.3 Middle CASE			
1.4.4 Lower CASE			
1.4.5 Post CASE			
1.5 Kontrola konzistence modelů v projektu			
1.6 Možná úprava uživatelského rozhraní			
1.7 Podporuje meta-CASE?			
1.8 Je schopný definice vlastního diagramu?			
1.9 Reverzní engineering pro používané jazyky?			
1.10 Přidání vlastních prvků			
1.11 Verzování modelů			
1.12 Sdílení modelů mezi uživateli			
1.13 Rozlišení uživatelů, uživatelská práva			
2. Komunikace, rozhraní			
2.1 Podpora komunikace se současnými (plánovanými) aplikacemi			
2.2 Podpora současného (plánovaného) databázového stroje			
2.3 Exporty do proprietálních datových formátů – XML, apod.			

Ukazatele / Hodnotící kritéria	*Váha ukazatele*	*Požadované minimum*	*Hodnocení*
2.4 Vytváření reportů			
2.5 Generování dokumentace			
2.6 Importy z a exporty do progr. jazyků – Java archiv (.jar), .NET (.exe, .dll), apod.			
2.7 Export modelu do sekvence databázových příkazů			
3. Podpora			
3.1 Školení uživatelů			
3.2 Zákaznický servis			
3.3 Podpora při problému – telefon, mail, diskuze, dojezd do firmy			
3.4 Pozáruční servis			
3.5 Politika update a upgrade			
4. Technické požadavky			
4.1 Síťová správa			
4.2 Zálohování repozitoře			
4.3 Požadovaný operační systém			
4.3.1 Stanice			
4.3.2 Server			

Během vytváření hodnotící tabulky s požadavky na CASE nástroj je tedy na manažerovi, aby pro všechny porovnávané produkty stanovil váhy jednotlivých ukazatelů, podle kterých bude dále prováděno hodnocení vybíraných nástrojů. Vzhledem k maximální objektivnosti, je nejvhodnější, když hodnota „Váhy ukazatele“ nebude uvedena v materiálech (hodnotících tabulkách), které budou použity v průběhu analýzy jednotlivých nástrojů (viz etapa analýza trhu), a to i za předpokladu, že analýzu provádí stejná osoba, která navrhla poměr vah jednotlivých ukazatelů. Důvodem je zpětná dohledatelnost a dokladovatelnost výběru.

Úspěšné zakončení této etapy předpokládá vyplněnou hodnotící tabulku vzhledem k unikátním potřebám konkrétní firmy (rozumějme tabulku s vyplněnou definicí vah jednotlivých ukazatelů a s definováním požadovaných minimálních hodnot) spolu s tabulkou definic hodnocení jednotlivých kritérií.

13.6. Sběr dat a hodnocení aktuální situace na trhu s CASE nástroji

V této etapě dochází k získání technických informací a informací o nástrojích bez samotného uživatelského kontaktu s nimi. Tato a příští etapa slouží pouze k výběru omezeného vzorku nástrojů, se kterými se bude dále pracovat. Informace získané v rámci této etapy jsou subjektivním hodnocením nástrojů podle kritérií v hodnotící tabulce a to na základě volně dostupných informací.

V rámci této metodiky jsem stanovila dvě metody hodnocení CASE nástrojů. Každá z nich má své výhody a nevýhody, je tedy pouze na manažerovi, k jaké z nich se přikloní.

- Metoda maximální verze – v rámci této metody se provádí hodnocení CASE nástrojů pouze z hlediska maximální získatelné verze. Pro každý CASE nástroj se provede pouze jedno hodnocení na základě nejvyšší verze, která je v daný moment k dispozici. Výhodou je rychlejší a přehlednější zpracování souboru informací o CASE nástrojích, zvláště pokud má nástroj složitější management různých volitelných modulů. V tom případě by byla nutnost pokrýt veškeré jejich kombinace. Nevýhodou pak je možná vysoká náročnost na vybavení u maximální verze, která nemusí odpovídat náročnosti u nejnižší možné verze nástroje, která by plně pokryla požadavky společnosti. Rizikem je nepřesnost nezachytitelná při širším vyhodnocení nasbíraných údajů.
- Metoda komplexních verzí – v rámci této metody se provádí hodnocení všech verzí vyhodnocovaných CASE nástrojů. Výhodou je komplexní zmapování trhu a možností, následné jednodušší vyhodnocení výběru. Citelnou nevýhodou je obtížnost hodnocení různých kombinací verzí u jednotlivých nástrojů, což způsobuje zvýšené časové a tím pádem i finanční a personální náklady na průběh projektu.

Otevřenou otázkou zůstává možnost použití dynamické kombinace těchto dvou metod, podle typu jednotlivých nástrojů. Zde je ale nutné mít na mysli nárůst případné nepřesnosti při výběru optimálního nástroje.

Získání potřebných informací v této etapě lze pomocí několika v praxi použitelných postupů. Ráda bych jich několik nastínila a shrnula jejich výhody a nevýhody. Pro manažera tento přehled může představovat vodítko pro vlastní iniciativu, případně možnou kombinaci postupů níže uvedených.

- Zjištění informací z internetových stránek výrobce / produktu - tato metoda bude zřejmě nejvýhodnější pro menší firmu, která nemá natolik velkou kupní sílu, aby se o ni zajímal přímo prodejce a obstaral jí přímo požadované informace. Výhodou je relativní rychlost a nenáročnost na komunikaci. Nevýhodou je nutnost orientace v různých nabídkách a prezentacích, nebezpečí nerelevantních či nedostupných informací, vyšší časová náročnost, velké nebezpečí nepostihnutí všech (nebo alespoň většiny) produktů na trhu s CASE nástroji a to zejména pokud má firma zájem o nenáročné řešení, neboť poté je otázkou, zda najde například vyhovující volně šiřitelný nástroj.

- Obeslání výrobce/prodejce s žádostí o informace o produktu - touto metodou lze jednoduše získat dostupné informace o produktech, případně s upřesňujícím dotazem i informace o neuváděných unikátních vlastnostech nástroje. Tato metoda je využitelná pro firmu, která uvažuje o nasazení CASE nástroje na středním a větším počtu počítačů nebo má zájem o komplexní řešení. Výhodou je nízká náročnost na zaměstnance firmy, vysoká komplexnost získané informace. Vzhledem k tomu, že vycházíme z oslovení výrobců firmou, je shodně jako v předchozím postupu nevýhodou nebezpečí nepostihnutí všech nástrojů.
- Využití portálu zajišťující B2B[153] komunikaci a zadání poptávky po CASE nástroji - touto metodou lze jednoduše oslovit velké množství výrobců a prodejců CASE nástrojů. Výhodou je, že výběr není omezen znalostí hledajícího zaměstnance nebo na schopnost SEO optimalizátora jednotlivých stránek a vyhledávače. Navíc reagující firma je ochotná dodat podklady k jí nabízeným case nástrojům- Nevýhodou je, že kontaktovat firmu nemusí přímo výrobce, ale překupující firma, která nástroj nabídne za cenu vyšší, než je ta, za kterou lze nástroj od výrobce běžně získat. Další nevýhoda je na některých serverech pak platba za vyhledanou transakci a obecná problematičnost při rozhodování o vítězi na základě této metodiky.

Ráda bych upozornila na nutnost zveřejnění výběrového řízení na B2B (resp. G2B[154]) portálu podle zákona 137/2006 Sb. o veřejných zakázkách[155], pokud je firmou úřad státní správy, samosprávy nebo státní podnik.

Zápis hodnocení vychází z podkladů vytvořených v předchozích etapách. Hodnotitel nebo hodnotitelé vychází z hodnotících tabulek a legendy, které jim byly zprostředkovány. Na základě dostupných údajů zapisují maximálně objektivní hodnocení nástroje (případně konvertují zjištěný stav na bodovou hodnotu hodnocení). V případě, že se na hodnocení bude podílet více osob, existuje nebezpečí, že i přes snahu o objektivní hodnocení, dojde ke zkreslení zapisovaných hodnot do hodnotící tabulky. Možné řešení je nechat dostatečně velkou část (cca 10%) hodnocených nástrojů hodnotitelům překrývat, na základě porovnání hodnocení u CASE nástrojů hodnocených více hodnotiteli, lze následně upravit hodnocení zpracovatele vzhledem k jeho směřující tendenci.

Pokud jsme vyšli z výchozí situace, kdy je ve firmě již nevyhovující CASE nástroj a chceme jej tedy podrobit porovnání s jinými nástroji, je nutné, aby stejnou metodou, jakou hodnotíme ostatní nástroje, zhodnotili i doposud používaný CASE nástroj.

---

[153] B2B je zkratkou pro "business-to-business" a znamená, že jeden podnik dělá business s jiným podnikem.

[154] B2G je zkratkou pro „business-to-government“ a znamená, že podnik dělá business s vládou.

[155] Zákon o veřejných zakázkách č. 137/2006 Sb. [online]. 2008. Dostupné na: http://www.verejna-zakazka.cz/zakon/

Etapa je považována za ukončenou, když je pro všechny vybrané nástroje vyplněná hodnotící tabulka, případně když jsou u všech nástrojů vyplněna i nově přidaná hodnocení.

13.7. Širší vyhodnocení

Cílem této etapy je prostý výběr skupiny CASE nástrojů, které nejvíce vyhovují původnímu zadání, s pomocí jednoduchých aritmetických postupů.

V předchozí etapě jsme si stanovili, že sloupec „Požadované minimum" slouží k vyřazení zcela nevyhovujících nástrojů, zatímco sloupec „Váha ukazatele" slouží pro porovnání obecně vyhovujících nástrojů za účelem nalezení nevhodnějších. Pro další zpracování je v této etapě nutné provést kontrolu a konsolidaci nasbíraných dat, jelikož se z nich bude po celou dobu výběru vycházet.

V případech, kde není nastavené žádné požadované minimum (kritérium bylo ignorováno, označeno jako nedůležité), se pro potřeby dalšího zpracování bere jako nastavené minimum na hodnotu nula.

Nejprve je potřeba vyřadit ty nástroje, které v jakémkoli hodnotícím kritériu nedosáhly stanoveného minima. To se provede tak, že se vyřadí každý nástroj, který má v jakémkoliv řádku tabulky ve sloupci „Hodnocení" nižší hodnotu než je hodnota uvedená ve sloupci „Požadované minimum". Tímto dostáváme ze souboru všech nástrojů pouze ty, které vyhovují požadavkům firmy.

Vyhodnocení pokrytí požadavků se provede podle následující metodiky: v každém řádku se provede vynásobení váhy ukazatele a jeho dosažené hodnoty. Výsledná čísla ze všech řádků se následně sečtou a součet představuje index úspěšného naplnění požadavků firmy.

Do další etapy postupují pouze nástroje na prvních 3 - 5 místech. Toto rozmezí je dáno tím, že ve výsledcích může dojít ke shlukování nástrojů s podobným indexem, kdy například mezi druhým a pátým místem je menší rozdíl, než je výrazný rozdíl mezi pátým a šestým místem. V tomto případě do užšího výběru postoupí prvních pět nástrojů. Viz níže uvedený obrázek možného případu grafického uspořádání.

Obr. 35: Případ shlukování nástrojů s podobným indexem

V případě, kdy je již ve firmě používán CASE nástroj, který je označen jako nevyhovující, je nutné přidat mezi okruh nástrojů postupujících do další etapy i výsledky stávajícího CASE nástroje.

V případě použití metody komplexních verzí, se z každého CASE nástroje uvažuje pouze ta verze, která se umístila nejvýše, tj. měla nejvyšší hodnotu vypočteného indexu.

Výstupem etapy je soubor odhadem čtyř nejlépe hodnocených CASE nástrojů, které postupují do užšího hodnocení.

13.8. Užší hodnocení

V této etapě již dochází k přímému porovnání předvybraných CASE nástrojů. Současně do hodnotících kritérií začíná vstupovat cena nástroje a praktické zkušenosti uživatelů. Doposud nebylo nutné rozebrat jednotlivé ukazatele a úspěšnost, s jakou CASE nástroje splnily předpoklady v jednotlivých skupinách. Toto hodnocení již k tomuto přikládá význam.

V této etapě již doporučuji kooperaci se zaměstnanci, protože v rámci užšího výběru je již možnost seznámit zaměstnance a vedoucí projektu s jednotlivými nástroji. Posuzuje se jejich ergonomii[156] ovládání, rozhraní, uživatelskou přívětivost, rychlost zpracování a využití této osobní zkušenosti k nalezení CASE nástroje, který bude do firmy nasazen.

Ve srovnání s numerickým přístupem (týkající se širšího hodnocení), je užší hodnocení již mnohem méně omezující. Rozdíly z hlediska kvality produktu jsou již minimální a z tohoto důvodu tedy nemá moc smysl již porovnávat globální souhrny vlastností. Principem této etapy je porovnat CASE nástroje ze dvou hledisek:

1) zhodnocení mechanismu práce s CASE nástrojem, při kterém lze velmi dobře získat zkušenosti od budoucích uživatelů nástroje, jejich preference, apod.,

2) zhodnocení nákladů na pořízení CASE nástroje.

V případě, že dosud byla použita metoda maximální verze, pak do porovnání vstoupí za každý CASE nástroj tato maximální verze a verze, která jako nejlevnější plně podporuje požadavky na CASE nástroj.

V případě použití metody komplexních verzí, do porovnání vstupuje verze, která se umístila nejvýše v předchozí etapě (a tedy verze, kvůli které se do této etapy CASE nástroj vlastně dostal) a verze, která jako nejlevnější plně podporuje požadavky na CASE nástroj.

Setřídíme-li všechny takto získané verze dle nákladů na pořízení, dostáváme první dimenzi grafu, na kterém nalezneme vítězný CASE nástroj.

Získání ohlasu od uživatelů na kvalitu práce s nástrojem je možné provést několika způsoby. Metoda se zřejmě bude lišit podle velikosti firmy, resp. odhadovaných nákladů na pořízení (ve formálním vyjádření zisku dodavatele). Hodnocení provádíme na základě:

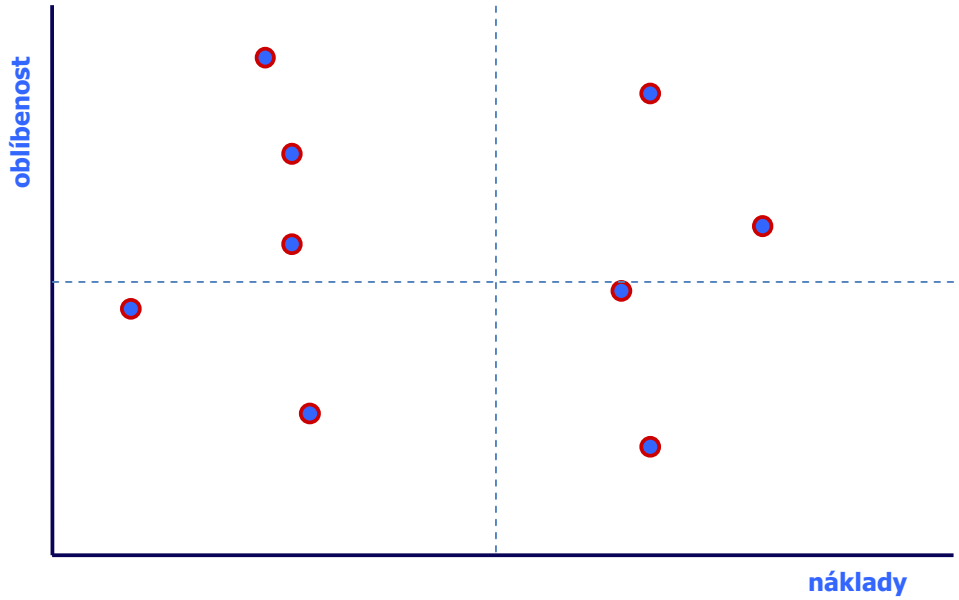
[156] Ergonomie je obor, který se zabývá studiem vztahů mezi člověkem a technickými systémy, které člověk vytváří.

- zkušeností na základě předváděcí demonstrační verze – která je nainstalovaná přímo ve firmě z volně dostupných zdrojů nebo na základě zaslané demonstrační verze dodavatelem,
- zkušenost na základě workshopu – krátkodobého semináře u dodavatele, který zaměstnance firmy jako potencionální zákazníky, provádí použitím CASE nástroje.

Získané zkušenosti s CASE nástrojem mohou zaměstnanci vyjádřit například vytvořením pořadí nástrojů dle oblíbenosti. Během přepočtu dostane za každou první pozici CASE nástroj plný počet bodů, za každou druhou o jeden bod méně, až na poslední pozici dostane jeden bod. Součet bodů získaných v této etapě tvoří druhou dimenzi grafu.

Nyní je již možné do grafu vynést na souřadnice tvořené kombinací nákladů na pořízení nástroje a bodového hodnocení uživatelů interpretaci CASE nástroje a provést vyhodnocení.

Obr. 36: Vytvoření grafu na základě vyhodnocených informací



Z obrázku vyplývá, že optimální CASE nástroj nalezneme v oblasti s nejvyšším hodnocením uživatelů a nejnižšími náklady. Za předpokladu, že se v této oblasti nenachází žádný nástroj, má manažer na výběr, zda upřednostní nástroje, které se umístily v oblasti s vyššími náklady na pořízení, nebo ty, které byly hůře hodnoceny zaměstnanci. Toto rozhodnutí může být ovlivněno rozpočtem na pořízení CASE nástroje a dalšími skutečnostmi. V případě, že se

všechny nástroje seskupily v oblasti drahých a nízko hodnocených, doporučila bych než vybírat „menší zlo“ spíše zkusit přibrat do užšího výběru první až druhý CASE nástroj, který nepostoupil v širším výběru a provést kroky definované v užším výběru i pro něj. Pokud by ani poté nedošlo k přijatelnému výsledku, doporučuji provést revizi průběhu výběru a případně upravit váhy hodnocení.

V situaci, kdy podnik již CASE nástroj vlastní, ale není s ním spokojen, přidáme k vlastnostem nástrojů, které v této etapě porovnáváme, také maximální verzi a verzi, která jako nejlevnější plně podporuje požadavky na CASE nástroj. V případě, že žádná taková verze u stávajícího nástroje neexistuje, tj. ani maximální verze nepodporuje plně dané požadavky, nebudeme ji přidávat vůbec. Skutečnost, že aktuální CASE nástroj bude mít výhodnější finanční náklad na pořízení je více než předvídatelné.

Na konci této etapy máme již vybraný optimální CASE nástroj pro firmu a je tedy možné se obrátit na dodavatele se zakázkou. Touto etapou popis metodiky končí. Zavedení CASE nástroje je sice součástí projektu, ale je již mimo oblast tématu této práce.

13.9. Kooperace a komunikace se zaměstnanci

Vzhledem k tomu, že úspěšnost projektu závisí na tom, zda zaměstnanci budou ochotni nový nástroj používat a budou schopni jej využívat efektivně, vytvořila jsem za tímto účelem jakousi uměle vytvořenou etapu složenou ze složky kooperace a komunikace se zaměstnanci. Mimo ostatní etapy je tato etapa také umístěna proto, že provedení jednotlivých úkonů v rámci této etapy lze posouvat dle potřeby během průběhu projektu.

Zodpovědnost za komunikaci se zaměstnanci má typicky manažer projektu. Na počátku projektu je vhodné mezi zaměstnanci identifikovat jednotlivé skupiny[157] dle postojů k nově nasazovanému CASE nástroji:

- oponenti, kteří obecně zaujímají negativní postoj a je nutné zjistit, zda je tento postoj zapříčiněn novým nástrojem jako takovým či odporem ke změně. Takovou skupinu je zapotřebí objevit a neutralizovat její postoje, například jejich zapojením do procesu výběru CASE nástroje, nebo zapůsobit na ně pomocí skupiny otevřených příznivců.
- skrytí oponenti neboli oportunisti, jsou ti, kteří zastávají negativní postoj, ale zvenku projevují podporu pro uskutečnění projektu. Jsou hůře identifikovatelní, doporučení na práci s nimi je stejné jako s oponenty otevřenými, bohužel je složitější je zjistit nebo odhalit.
- skupinou, o kterou je třeba pečovat, jsou otevření příznivci. Tito lidé jsou hybnou silou, na kterou je vhodné působit ve fázi sběru interních informací. Tito lidé tvoří základ týmu pro užší hodnocení CASE nástrojů a dají se využít jako propagátoři nově vybíraného CASE nástroje.

[157] Sodomka, P. Investujte a vydělejte. BIZ, Computer Press Brno 2007, ISSN ISSN: 1213-063X

- poslední skupinou jsou potenciální příznivci, kteří nezaujímají ani pozitivní ani negativní postoj ke změně. Nejsou tedy ještě plně přesvědčeni o prospěšnosti plánované změny, a proto je vhodné se zaměřit na získání jejich přízně.

V jednotlivých skupinách je možné v průběhu etap provádět různé akce s důrazem na zvýšení povědomí o projektu, nastavení pozitivního přijetí vybíraného CASE nástroje, atd.

Úkolem této etapy je mimo jiné také to, aby došlo k přijetí CASE nástroje na všech úrovních zaměstnanců firmy, kteří ho ke své práci potřebují. K tomu je třeba rozpoznat, ze které úrovně firmy přišel původní impuls k zavedení CASE nástroje:

- ze strany samotných zaměstnanců – vývojářů a analytiků, kteří budou s CASE systémem pracovat,
- ze strany nižšího managementu - jako je teamleader, project manager, kteří mají přímý vliv na zaměstnance s tvořivou činností a zodpovědnosti za projekty a splnění plánů,
- ze strany středního managementu - což ve větší firmě bývá exekutiva a rozpracování rozhodnutí na globální úrovni (možným impulsem může být zjištění, že se původní CASE nástroj vlastně vůbec nepoužíval).

Pro úspěšné zavedení a používání CASE nástroje ve firmě je nutné, aby byly upokojeny zájmy všech tří výše uvedených skupin uživatelů a všechny tři skupiny tento projekt braly jako přínos. Dá se předpokládat, že skupina, která prosazuje zavedení CASE nástroje, již jeho přínos vidí, a tedy není nutné ji tolik stimulovat dalšími akcemi. Samotná potřeba seznámení zaměstnanců s projektem a vytvoření pozitivní vazby k nasazovanému systému[158] je mimo jiné jedním z klíčových prvků úspěšného využití CASE systému v praxi. Veškeré popisované kroky mají za cíl zvýšit nejen informovanost, ale také motivaci zaměstnanců, aby byl nástroj používán. Toto může urychlit a zjednodušit jeho integraci mezi portfolio používaného podnikového software.

Následující činnosti mohou použít jak jednotlivci, tak zainteresované skupiny:

- obeznámení zaměstnanců se záměrem – typicky se používá v případě, kdy došlo k rozhodnutí o pořízení CASE nástroje na úrovni nižšího nebo středního managementu. Tento nástroj umožní skupině zaměstnanců na nižší úrovni obeznámení s projektem,
- obeznámení zaměstnanců s výhodami nasazení CASE nástroje do skupiny firemních aplikací – tyto výhody mohou být různé, závisí na tom, komu jsou reprezentovány. Řadoví zaměstnanci upřednostní ulehčení práce, přístup k informacím, nižší management preferuje zvýšení produktivity, zálohu znalostí, snížení chybovosti, střední management uvítá snížení nákladů na vývoj.
- zapojení zaměstnanců do rozhodování o výběru nového CASE nástroje – tato akce je klíčová zejména ve fázi sběru interních informací a ve fázi užšího

[158] Lending, D. Chervany, N. L. Case tool use and job design: a restrictivness/flexibility explanation, The Journal of Computer Information System, Fall 2002, str. 81 - 90

výběru. Velmi často díky zapojení budoucích uživatelů nástroje lze získat cenné připomínky nebo doporučení, které slouží k výběru optimálního nástroje. Další výhoda tohoto kroku je, že vytváří ztotožnění se budoucích uživatelů s vybraným nástrojem.

- školení a představení možností produktu či produktů – školení je metoda, jak současně zajistit efektivní využívání CASE nástroje a současně zvýšit povědomí o nástroje a zájem na jeho používání při práci. Školení lze uskutečnit v obecné formě, pouze jako školení modelovacích technik podle určité metodiky, nebo může jít o školení v používání vybraného nástroje - poté ale musí proběhnout až po výběru CASE nástroje a dodavatele.

Tab. 5: Doporučené aktivity pro podporu přijetí CASE nástroje

Etapa \ Aktivita	***Obeznámení zaměstnanců se záměrem***	***Obeznámení zaměstnanců s výhodami nasazení***	***Zapojení zaměstnanců do rozhodování o výběru***	***Školení a představení možností produktu***
Zjištění potřeby nového CASE nástroje	✔	✔		
Sběr interních informací pro výběr CASE nástroje	✔	✔	✔	
Definování požadavků na CASE nástroj				
Sběr dat a hodnocení aktuální situace na trhu s CASE nástroji				
Širší vyhodnocení				
Užší vyhodnocení			✔	
Zavedení				✔
Běžný provoz CASE nástroje				✔

13.10. Outsourcing výběru

Outsourcing výběru CASE nástroje na poradenskou firmu je velmi jednoduše proveditelným řešením. Hlavní výhody[159] outsourcing výběru se dají shrnout do následujících bodů:

- zadavatelská firma se může soustředit pouze na předmět podnikání, není nutné štěpit pozornost zaměstnanců a zvyšovat náklady směrem k výběru

[159] Synek, M. Manažerská ekonomika. Grada Publishing, Praha 2000, 4. vyd. ISBN 80-. 247-9069-9. str. 204

CASE nástroje. Firma se tak může věnovat jen a pouze svým core procesům a dělat je tak co nejlépe,

- outsourcing přináší zrychlení procesu výběru, protože odpadá potřeba získání znalostí a orientace se v problematice,
- u poradenské firmy se předpokládá vyšší znalost problematiky, rozšířené obzory v popisu možností výběru vhodného software, aktuálních dat a zkušeností z podobných projektů,
- poradenská firma je obeznámena s aktuální situací na trhu CASE nástrojů, jeho trendy a má schopnost doporučení vhodného nástroje v tomto směru,
- poradenská firma je schopna poskytnout, během celého procesu výběru a zavádění podporu zcela nezávisle na personálních možnostech objednatele. Objednatel nemusí řešit personální otázky spojené se specifickými znalostmi dané oblasti, po celou dobu projektu má záruku, že dostane stejně kvalitní služby.

S nespornými přínosy outsourcing musíme vzít v úvahu také rizika a nevýhody s ním spojené. Vybrala jsem takové nevýhody, které jsou relativně pravděpodobné, či takové, které ohrožují samotný úspěch projektu, případně se jedná o vysoce problematické záležitosti pro celou firmu nezávisle na její velikosti. Zásadními výhodami je především:

- nerozpoznání, nebo špatné rozpoznání specifických potřeb zadavatele, nebo nevhodné stanovení jejich stupně závažnosti, které vychází z relativní unikátnosti každého projektu a snižuje se s počtem již absolvovaných projektů poradenskou firmou,
- vzhledem k nutnosti těsného svázání zadavatele s poradenskou firmou z důvodu přesného uchopení potřeb a zvyků zadavatele, je případné vyzrazení vnitřních firemních procesů, jejího know-how, best practices a ostatních náležitostí, které ji umožňují vedoucí nebo unikátní postavení na trhu, mezi to se počítá i znalost specifik zákazníků, pro které objednatel může vytvářet aplikace,
- může se objevit případ, kdy je poradenská firma úzce svázána s jedním dodavatelem řešení, které je pak nabídnuto jak nejvýhodnější pomocí např. úpravy koeficientů výhodnosti během etapy vyhodnocení požadavků zadavatele.

Pokud bychom hodnotili, zda využít outsourcing pro výběr CASE nástroje, je nutné si uvědomit, že můžeme či naopak nemusíme dosáhnout úspor financí, jelikož každá firma bude mít jiné náklady týkající se výběru správného CASE nástroje a tudíž se jí outsourcing může a také nemusí vyplatit (stejně tak může mít jinak kvalifikované zaměstnance, kteří svými znalostmi mohou převyšovat znalosti specializovaných firem). Proto nelze toto hledisko brát jako doporučení pro outsourcing či jeho zamítnutí. Outsourcing výběru si znázorníme na následujícím diagramu.

Obr. 37: Outsourcing výběru

Uvedený diagram znázorňuje tmavou barvou etapy, které jsou plně v kompetenci poradenské firmy, tedy jsou zpracovávány mimo zadavatele bez nutnosti investování jeho zdrojů. Jedná se o plně přenosné činnosti, na jejichž provádění není potřeba primárně dohlížet.

Světle modrým tónováním jsou označeny etapy, kde je nutná kooperace poradenské firmy se zadavatelem. Jedná se zejména o počáteční fázi, kdy zadavatel předává své požadavky a potřeby zpracovateli, provádí se mapování prostředí, do kterého se bude CASE systém zavádět a další činnosti uvedené výše v popisu etapy. Opačný tok informací se očekává ve finální etapě výběru, kdy naopak zpracovatel zdůvodňuje doporučení výběru, podílí se na workshopech s uchazeči, atd. Zkušenosti poradenské firmy mohou také ovlivnit komunikační hledisko projektu, zejména znalostí potřeby komunikace se zaměstnanci zadavatele. Unikátní možností, jak spolu se zástupci poradenské firmy informovat zaměstnance zadavatele o plánovaném či spouštěném projektu, je tzv. kick-off meeting, který je doporučen systémovými integrátory[160] jako standardní počátek projektu, na kterém dochází k představení projektu, jeho odůvodnění a zdůraznění výhod pro zaměstnance, zvýšení motivace zaměstnanců ke kooperaci atd.

Výstupy z jednotlivých etap, jak jsem je určila výše v této kapitole, je pro zadavatele vhodné zohlednit jako kontrolu správnosti postupu v jednotlivých etapách, a to alespoň po ukončení každé z etap. Kontrolou by měla procházet faktická správnost, zda materiály odpovídají realitě a požadované jakosti, ale také fakt, zda se poradenská firma neodchyluje od požadavků a potřeb zadavatele.

O předání výsledků by se měla vést dokumentace, stejně jako o celé korespondenci mezi zadavatelem a zhotovitelskou poradenskou firmou a to na obou stranách. Důvodem může být snaha o předejití dezinformace, upřesnění termínů a plánů, nebo z důvodu případné pozdější dokladovatelnosti pro případ problémů.

Jedním z nejvíce zvažovaných faktů, mluvících pro outsourcing výběru CASE nástroje, je skutečnost, že firma která již nasazovala určitý CASE nástroj a která je v kontaktu s jeho uživateli, tímto získala zkušenosti s dodavatelem tohoto nástroje. Získala tedy určité informace o kvalitách jeho proklamovaných služeb, a další informace, které nejsou zřejmé z hodnocení výrobku, čímž ovlivňuji komfort jeho užívání. Tyto informace jsou bohužel hůře přenositelné a na trhu hůře dostupné a pracovníci zadavatele se s nimi často nemají ani možnost seznámit.

13.11. Co je třeba neopomenout

Sebelepší a sebekvalitnější CASE nástroje nám nezaručí z ničeho nic stoprocentní úspěšnost po jejich pořízení, neboť se jedná „pouze“ o nástroje,

[160] Kick-off meeting: Základ dobré organizace implementace [online]. essence business solutions 2008. Dostupné na: http://www.essencebs.cz/cz/jak-vedeme-projekty/2-kick-off-meeting

které mají za úkol zautomatizovat a tím pádem usnadňovat lidem činnosti, které jsou bez využití těchto nástrojů, řekněme manuálně“ velmi náročné.

Je třeba neopomenout, že každá firma má jinou organizační strukturu, odlišnou úroveň zralosti jednotlivých procesů (viz. již zmíněné CMM v podkapitole 8.5.2.1), různou organizaci řízení průběhu vývoje projektu a odlišný charakter a rozsah projektů a jiné nároky a požadavky na řízení projektů.

Proto je nejdůležitější částí celého procesu výběru CASE nástroje primární (prvotní) analýza, která nejprve zváží, zda je zapotřebí tento nástroj skutečně pořizovat (čímž nechci jeho nesporné přínosy nijak zlehčovat), neboť ne každá firma jej ve skutečnosti opravdu potřebuje. Je třeba zvážit podmínky konkrétní organizace, jaké projekty realizuje, jaké má organizace záměry a očekávání od daného nástroje. Výše uvedená metodika by nám při správném přizpůsobení dané firmě, kterou její obecnost umožňuje, měla pomoci vyhnout se situaci, kdy dochází ke zmařené investici, neboť daný nově pořízený systém či nástroj není vůbec užíván.

13.12. Závěr

Výše uvedenou metodiku řadím mezi méně restriktivní, neboť předkládá doporučení při volbě metod pro splnění jednotlivých etap popisovaných touto metodiku. Soubor metod a postupů je odvozený z praxe a best practices získané z dostupných zdrojů a jiných metodik zabývajících se podobným tématem.

Proces výběru nástroje jsem rozčlenila do jednotlivých bloků, jmenovitě zjištění potřeby nového CASE nástroje, sběr interních informací pro výběr CASE nástroje, definování požadavků na CASE nástroj, sběr dat a hodnocení aktuální situace na trhu s CASE nástroji, širší vyhodnocení, užší vyhodnocení, zavedení, běžný provoz CASE nástroje a etapa kooperace a komunikace se zaměstnanci, k nimž jsem sestavila doporučení, která by se při výběru měla zohlednit.

Metodika rozebírá dvě výchozí situace:

1) daná organizace již určitý CASE nástroj vlastní,

2) firma dosud CASE nevyužívala.

V prvním případě metodika poskytuje přehled možných důvodů neúspěchu při používání aktuálního modelovacího nástroje a nabízí možná řešení bez nutnosti pořízení nového modelovacího nástroje. V druhém případě uvádí metodika doporučený postup průběhu výběru.

Metodika se nezabývá procesem implementace daného CASE nástroje, ale neopomíjí kooperaci a komunikaci se zaměstnanci, které jsou procesy doprovázejícími proces výběru od fáze sběru až po etapu užšího hodnocení včetně. Doporučuje identifikaci jednotlivých rolí pracovníků podle jejich postoje k nově nasazovanému CASE nástroji, jejíž uvědomění nám umožní lepší

zaměření na tyto skupiny a tím i přijetí CASE nástroje na všech úrovních zaměstnanců firmy, kteří ho ke své práci potřebují. Metodika uvádí doporučené aktivity pro podporu přijetí nástroje.

V neposlední řadě se metodika zaměřuje i na možnost outsourcingu výběru nástroje a poskytuje doporučení pro získání správných výstupů pro outsourcingovou firmu a správnou kooperaci s ní.

Metodika zdůrazňuje, že je nutné si uvědomit odlišnost organizačních struktur, charakteru a rozsahu projektů, rozdílné nároky a požadavky na řízení projektů a také rozlišnou úroveň zralosti jednotlivých procesů. Proto je třeba brát výše uvedenou metodika pouze jako doporučení, která si musí daná firma přizpůsobit svých potřebám.

14. Závěr

V diplomové práci jsem se zabývala vytvořením metodiky pro výběr vhodného CASE nástroje. Tyto softwary dnes hrají nezastupitelnou roli při vývoji a návrhu informačních systémů a jen velmi těžko si tyto činnosti lze představit bez technologie CASE. Na trhu existuje nepřeberné množství nástrojů a najít ten nejvhodnější se stává složitým úkolem.

Cílem práce bylo vytvoření obecné metodiky pro výběr a nasazení CASE nástroje. Tento cíl se mi podařilo splnit a metodika je hlavním přínosem mé práce. Její výhodu spatřuji především v široké přizpůsobitelnosti. Metodika nabízí doporučení pro jednotlivé etapy výběru a nasazení CASE. Proces výběru nástroje jsem rozčlenila do jednotlivých bloků, jmenovitě zjištění potřeby nového CASE nástroje, sběr interních informací pro výběr CASE nástroje, definování požadavků na CASE nástroj, sběr dat a hodnocení aktuální situace na trhu s CASE nástroji, širší vyhodnocení, užší vyhodnocení, zavedení, běžný provoz CASE nástroje. V průběhu celého životního cyklu se kooperuje a komunikuje s budoucími uživateli, k čemuž jsem sestavila doporučení, která by se při výběru měla zohlednit.

V prvním případě, kdy se metodika zabývá situací, ve které již firma CASE nástroj vlastní, poskytuje přehled možných důvodů neúspěchu při používání aktuálního modelovacího nástroje a nabízí možná řešení bez nutnosti pořízení nového. V druhém případě, kdy firma dosud žádný CASE nástroj nepoužívala, uvádí doporučený postup průběhu výběru.

Metodika se nezabývá procesem implementace daného CASE nástroje, ale neopomíjí kooperaci a komunikaci se zaměstnanci, což jsou aktivity, které nutně doprovází proces výběru od fáze sběru až po etapu užšího hodnocení. Doporučuje identifikaci jednotlivých rolí pracovníků podle jejich postoje k nově nasazovanému CASE nástroji, což nám umožní lepší zaměření na tyto skupiny a tím i akceptaci CASE nástroje mezi zaměstnanci firmy, kteří ho ke své práci potřebují. Metodika uvádí doporučené aktivity pro podporu přijetí nástroje.

Metodika zdůrazňuje, že je nutné vzít v úvahu odlišnost organizačních struktur, charakter a rozsah projektů, rozdílné nároky a požadavky na řízení projektů a také různou úroveň zralosti jednotlivých procesů.

První kapitola se týká softwarového inženýrství, které se zabývá přístupem k vývoji, nasazení a údržbě softwaru a softwarové krize, neboť ta je jedním z milníků v historii softwarového inženýrství, díky kterému došlo k rozmachu modelovacích metodik a k vysoké penetraci CASE nástrojů.

Ve druhé kapitole jsem pro účely práce ujasnila nejdůležitější pojmy, zejména problémové termíny metodologie a metodika. Metodologie analyzuje principy a procedury tvorby a aplikace metodik a zkoumá způsoby řešení problémů. Metodika je již specifickým přístupem, definujícím doporučené postupy, principy, koncepty, dokumenty, metody, techniky a nástroje pro vývoj IS.

Ve třetí kapitole jsem řešila životní cyklus vývoje IS, který se skládá z jednotlivých etap pokrývajících celý vývoj IS, jmenovitě zachycení požadavků na systém, tvorba konceptuálního a implementačního modelu, implementace a zavedení, testování, údržba systému a jeho provoz a stažení systému z užívání.

Podívala jsem se blíže na český standard ISVS 005/04, dále na mezinárodní standardy jako je ISO/IEC a 12207, ISO/IEC 15504.

Ve čtvrté kapitole jsem se zaměřila na metodiky z hlediska vývoje, na jehož základě se metodiky člení na tradiční a agilní. Historickým přístupem metodik je tzv. tradiční přístup, jehož příkladem je klasická a Yourdonova metoda, Code and Fix model, metoda tunel, byrokratická metoda, Stagewise model, metoda vodopád (fontána), přírůstková metoda, spirálový model a model Win-Win. Tradiční metodikou je také model zralosti (SW-CMM model), metodika RUP a EUP, OPEN a PDIT. Agilní metodiky představují reengineering procesů pro budování IS/CT, zaměřují se na vývoj nového řešení, nikoliv na údržbu a provoz. Tyto metodiky pocházejí z poznání, že jediný způsob, jakým lze ověřit správnost navrženého systému, je co nejrychleji jej vyvinout, předložit zákazníkovi a na základě zpětné vazby jej upravovat. Nejznámějšími příklady těchto metodik je Extrémní programování (XP), metodika SCRUM, Dynamic Systems Development Method (DSDM), metodiky Crystal, Feature Driven Development a Lean Development.

V páté kapitole jsem se věnovala již samotným CASE nástrojům. Tyto nástroje využívají metodik pro organizování, řízení a vývoj IS a to především rozsáhlých složitých projektů, které zahrnují mnoho softwarových komponent a na kterých spolupracuje velké množství lidí. CASE členíme na horizontálně orientované (pokrývají určitou či několik specifických částí životního cyklu) a vertikálně orientované (pokrývají všechny fáze životního cyklu) a dále je členíme podle životního cyklu na Pre CASE, Upper CASE, Middle CASE, Lower CASE, Post CASE a I-CASE.

V šesté kapitole jsem se zabývala metamodelováním a metainformačními systémy, neboť CASE nástroje se řadí mezi tzv. metasystémy. Zaměřila jsem se na metainformační systém MtS, který umožňuje popisovat, analyzovat a řídit IS/IT z pohledu všech významných dimenzí podle metodologie MMDIS.

V sedmé kapitole jsem se věnovala metodickému rámci IS/ICT (MeFIS), který je kolekcí metodických vzorů pro různé domény, typy řešení a způsoby řešení spolu s principy a procesy pro vytvoření konkrétní metodiky.

Osmá kapitola zkoumala aktuální situaci na trhu CASE nástrojů. Podrobným zkoumáním deseti vybraných nástrojů, jsem došla k závěru, že funkce nástrojů jsou v zásadních vlastnostech velmi podobné a jejich výběr je závislý na konkrétních potřebách dané firmy. Proto je zapotřebí metodiky, která usnadní jejich výběr.

Devátá, závěrečná, kapitola se zabývá již samotnou metodikou pro výběr a nasazení vhodného CASE nástroje. Vyhodnocovala jsem normu ČSN pro hodnocení a výběr CASE nástrojů (ČSN ISO/IEC 14102), která však pochází již z roku 1995 a její koncept je již poněkud zastaralý, např. předpokládá rozšíření schopností CASE do oblastí, které jsou dnes již samozřejmostí či naopak nejsou stále dosažitelné a posunuje některá hodnotící kritéria mimo dostatečně oblast dnešního zájmu a situaci na aktuálním trhu.

V práci jsem se nezabývala samostatnou oblastí project managementu, neboť již byla zpracována v řadě jiných prací a jedná se o samostatné téma.

15. Přílohy

15.1. Vlastnosti nabízené CASE nástroji

Tab. 6: Vlastnosti nabízené popisovanými CASE nástroji (více viz popis každého nástroje)[161]

Vlastnosti nástroje	***Select Enterprise 6***	***Rational Rose 2002***	***Oracle Designer 6***	***Power Designer 9***	***Win A&D***
metodiky	Select Perspective	Rational Unified Process	strukturované metodiky	objektové i strukturované	objektové i strukturované
notace	procesní hierarchický diagram, Diagram procesních řetězců, Use Case Diagram, Diagram tříd, Stavový diagram, Diagram sekvencí, Diagram spolupráce, Datový model, Obecná grafika	use case diagram, diagram sekvencí, diagram spolupráce, diagram tříd, stavový diagram, diagram komponent, diagram nasazení, diagram aktivit, datový model	Procesní model, Datové modely, Diagram datových toků, Diagram hierarchií funkcí, Konceptuální datový model, Klasický datový model, Multidimensionální model	Procesní model, Diagram tříd, Use case diagram, Sekvenční diagram, Diagram komponent, Diagram aktivit	Model tříd, Datový model, Diagram spolupráce, Diagram sekvencí, Procesní model (DFD), Model stavů, Stromový strukturní diagram, Model úloh, Diagram nasazení
fáze projektu	analýza, design, kódování	analýza, design, kódování	analýza, design, kódování	analýza, design, kódování	analýza, design, kódování

[161] Beneš, M. Přehled OO metodik a notací. Diplomová práce. VŠE 2005. kap. 3.3: Srovnání možností vybraných programů pro modelování

Vlastnosti nástroje	*Select Enterprise 6*	*Rational Rose 2002*	*Oracle Designer 6*	*Power Designer 9*	*Win A&D*
uživatelské modifikace	šablony generovaných dokumentů, vizualizace objektů	šablony pro generování dokumentů, frameworky projektů, skriptovací jazyk pro úpravu modelů (RoseScript), stereotypy	uživatelské objekty	šablony generovaných dokumentů, vizualizace objektů	šablony generovaných dokumentů, týmové požadavky
verzování	ano	ano (vazba na Rational ClearCase nebo MS Source Safe)	ano	ano	ano
kontrola správnosti	konzistence diagramů, tvorba na základě Select Perspective	konzistence diagramů, tvorba na základě RUP	konzistence	konzistence	konzistence
generování dokumentace	MS Word (výborná)	HTML (výborná)	výstupy o objektech v repository	HTML (výborná), RTF	HTML

Vlastnosti nástroje	*Select Enterprise 6*	*Rational Rose 2002*	*Oracle Designer 6*	*Power Designer 9*	*Win A&D*
generování databáze	Oracle, MS SQL Server, Interbase, ANSI SQL 92	DB2 (až v. 7), MS SQL Server (až v. 2000), Oracle (v 7, 8), Sybase (Adaptive Server 12) a ANSI SQL 92	Oracle, MS SQL Server, DB2, Sybase, ANSI SQL, ODBC	ANSI SQL, AS/400, DB2, Informix, Interbase, MS Access, MySQL a Postgre, MS SQL Server, Oracle (i verzi 9i), Sybase, Adabas, AllBase, Teradata, ODBC	Oracle, Sybase, SQL
reverse engineering z databáze	Ne. Na stránkách firmy LBMS (www.lbms.cz) povolené sítě) je k dispozici utilita napsaná ve Visual Basicu pro MS Excel pro reverzaci přes ODBC.	DB2 (až v. 7), MS SQL Server (až v. 2000), Oracle (v 7, 8), Sybase (Adaptive Server 12) a ANSI SQL 92	Oracle, MS SQL Server, DB2, Sybase, ANSI SQL, ODBC	ANSI SQL, AS/400, DB2, Informix, Interbase, MS Access, MySQL a Postgre, MS SQL Server, Oracle (i verzi 9i), Sybase, Adabas, AllBase, Teradata, ODBC	ne
generování kódu	C++, Java, MS VBasic, Delphi, PowerBuilder, Forte, Corba	ANSI C++, C++, Ada 83, Ada 85, CORBA, Java, Visual Basic, XML	Oracle Forms, PL/SQL, web PL/SQL	XML, Visual Basic, PowerBuilder, Java, CORBA, C++, C#, Analysis	C++, Java, Object Pascal, Delphi, Pascal, Fortran, Basic, C
reverse engineering kódu	C++, Java, MS VBasic, Delphi, PowerBuilder, Forte	ANSI C++, C++, CORBA, Java, Visual Basic, XML	Oracle Forms, PL/SQL, web PL/SQL	XML, Visual Basic, PowerBuilder, Java, CORBA, C++, C#, Analysis	C++, Java, Object Pascal, Delphi, Pascal, Fortran, Basic, C

Vlastnosti nástroje	*Select Enterprise 6*	*Rational Rose 2002*	*Oracle Designer 6*	*Power Designer 9*	*Win A&D*
uživatelské rozhraní	přehledné, docking oken	přehledné, docking oken	uživatelské poznámky, mnoho "wizards", méně přehledné a pohodlné	přehledné, docking oken, grafické synonymy objektů	graficky jednoduché, funkční
plusy	Select Perspective, generátor MS Word dokumentace, Process Mentor	RUP, frameworky, reverse/forward engineering, generování HTML dokumentace, rozsah modulů	DFD, silná vazba na implementaci databáze, tvorba data warehouse	podporované databáze, podporované jazyky, import z Rational Rose, HTML i RTF dokumentace	podporované metodiky, DFD, model obrazovek
mínusy	není reverse engineer z databáze		zaměření na databázové aplikace, potřebuje instalovanou databázi Oracle	není metodika tvorby IS	není metodika tvorby IS, není grafický use case diagram
generování kódu	C++, Java, MS VBasic, Delphi, PowerBuilder, Forte, Corba	ANSI C++, C++, Ada 83, Ada 85, CORBA, Java, Visual Basic, XML	Oracle Forms, PL/SQL, web PL/SQL	XML, Visual Basic, PowerBuilder, Java, CORBA, C++, C#, Analysis	C++, Java, Object Pascal, Delphi, Pascal, Fortran, Basic, C
reverse engineering kódu	C++, Java, MS VBasic, Delphi, PowerBuilder, Forte	ANSI C++, C++, CORBA, Java, Visual Basic, XML	Oracle Forms, PL/SQL, web PL/SQL	XML, Visual Basic, PowerBuilder, Java, CORBA, C++, C#, Analysis	C++, Java, Object Pascal, Delphi, Pascal, Fortran, Basic, C

Vlastnosti nástroje	*Select Enterprise 6*	*Rational Rose 2002*	*Oracle Designer 6*	*Power Designer 9*	*Win A&D*
uživatelské rozhraní	přehledné, docking oken	přehledné, docking oken	uživatelské poznámky, mnoho "wizards", méně přehledné a pohodlné	přehledné, docking oken, grafické synonymy objektů	graficky jednoduché, funkční
plusy	Select Perspective, generátor MS Word dokumentace, Process Mentor	RUP, frameworky, reverse/forward engineering, generování HTML dokumentace, rozsah modulů	DFD, silná vazba na implementaci databáze, tvorba data warehouse	podporované databáze, podporované jazyky, import z Rational Rose, HTML i RTF dokumentace	podporované metodiky, DFD, model obrazovek
mínusy	není reverse engineer z databáze		zaměření na databázové aplikace, potřebuje instalovanou databázi Oracle	není metodika tvorby IS	není metodika tvorby IS, není grafický use case diagram

15.2. Český standard pro náležitosti životního cyklu IS

Tab. 7: Vztah fází životního cyklu IS, strategických procesů a realizačních projektů IS[162]

Fáze životního cyklu IS	*Strategické procesy IS*	*Realizační projekty IS*
Příprava IS	• Definice potřeby IS • Příprava na zpracování nebo aktualizaci informační strategie IS • Příprava nebo aktualizace nástrojů strategického řízení IS	
Vývoj, provoz a údržba IS	• Tvorba a údržba informační strategie • Řízení bezpečnosti • Plánování a koordinace projektů • Plánování a řízení jakosti • Řízení požadavků a jejich monitorování	• Projekty akvizice • Projekty základního postupu vývoje • Projekty redukovaného postupu vývoje • Projekty provozu a údržby • Kombinované projekty
Ukončení činnosti IS	• Vyřazení	

Tab. 8: Průběh projektu akvizice[163] nechceš tu tabulku nechat vcelku?

Fáze projektu	*Standardem požadované povinné dokumenty*
1. Zahájení projektu	Projektový záměr - Evidenční list
2. Zahájení akvizice	Plán akvizice Systémové požadavky IS Projektová bezpečnostní dokumentace IS Úvodní studie IS
3. Příprava žádosti o nabídku	Žádost o nabídku

[162] Standard ISVS 005/02.01 pro náležitosti životního cyklu informačního systému [online]. Úřad pro veřejné informační systémy 2002. Dostupné na: http://www.mvcr.cz/micr/files/457/uvis_s005.02.01_v2002c5_20021218.pdf. str. 9

[163] Standard ISVS 005/02.01 pro náležitosti životního cyklu informačního systému [online]. Úřad pro veřejné informační systémy 2002. Dostupné na: http://www.mvcr.cz/micr/files/457/uvis_s005.02.01_v2002c5_20021218.pdf. str. 15

Fáze projektu	*Standardem požadované povinné dokumenty*
4. Příprava smlouvy a aktualizace	Smlouva
5. Monitorování dodavatele	Protokol o kvalifikačním testování systému Dokumentace vyžadovaná smlouvou
6. Akceptace a kompletace	Protokol o převzetí
7. Ukončení projektu	Ukončení projektu - Evidenční list

Tab. 9: Průběh projektu vývoje IS při použití základního postupu[164]

Fáze projektu	*Standardem požadované povinné dokumenty*
1. Zahájení projektu	Projektový záměr - Evidenční list
2. Zahájení vývoje	Plán vývoje Projektová bezpečnostní dokumentace IS
3. Analýza systémových požadavků	Systémové požadavky IS
4. Návrh architektury systému	Úvodní studie
5. Analýza softwarových požadavků	—
6. Návrh architektury softwaru	Globální návrh IS
7. Detailní návrh softwaru	Detailní návrh IS
8. Kódování a testování softwaru	—
9. Integrace softwaru	—
10. Kvalifikační testování softwaru	Protokol o kvalifikačním testování softwaru
11. Integrace systému	—
12. Kvalifikační testování softwaru	Protokol o kvalifikačním testování softwaru
13. Instalace	—

[164] Standard ISVS 005/02.01 pro náležitosti životního cyklu informačního systému [online]. Úřad pro veřejné informační systémy 2002. Dostupné na: http://www.mvcr.cz/micr/files/457/uvis_s005.02.01_v2002c5_20021218.pdf. str. 17

Fáze projektu	*Standardem požadované povinné dokumenty*
14. Podpora akceptace systému	Systémová příručka IS Uživatelská příručka IS Školící a učební texty Provozní bezpečnostní dokumentace Plán zavádění IS Protokol o převzetí IS
15. Ukončení projektu	Ukončení projektu - Evidenční list

Tab. 10: Průběh projektu vývoje IS při použití redukovaného postupu[165]

Fáze projektu	*Standardem požadované povinné dokumenty*
1. Zahájení projektu	Projektový záměr - Evidenční list
2. Zahájení vývoje	Plán vývoje IS Projektová bezpečnostní dokumentace IS
3. Analýza systémových požadavků	Systémové požadavky IS
4. Návrh architektury systému a softwaru	Návrh IS – redukovaný postup
5. Vývoj a integrace systému	—
6. Kvalifikační testování systému	Protokol o kvalifikačním testování systému
7. Instalace	—
8. Podpora akceptace systému	Systémová příručka IS Provozní bezpečnostní dokumentace Uživatelská příručka IS Školící a učební texty Plán zavádění IS Protok o převezení IS
9 Ukončení projektu	Ukončení projektu - Evidenční list

[165] Standard ISVS 005/02.01 pro náležitosti životního cyklu informačního systému [online]. Úřad pro veřejné informační systémy 2002. Dostupné na: http://www.mvcr.cz/micr/files/457/uvis_s005.02.01_v2002c5_20021218.pdf. str. 18 - 19

Tab. 11: Průběh projektu provozu[166]

Fáze projektu	*Standardem požadované povinné dokumenty*
1. Zahájení projektu	Projektový záměr - Evidenční list
2. Zahájení provozu	Plán vývoje IS Plán zavádění IS Plán údržby IS
3. Provozní testování	Systémová příručka IS Uživatelská příručka IS Provozní bezpečnostní dokumentace Školící a učební testy Zpráva o zavedení IS a konverzi dat Protokol o převzetí IS do provozního užívání - Evidenční list
4. Provoz systému	Provozní statistika Evidence poruch a mimořádných událostí
5. Údržba systému	Návrh migrace Zpráva o provedení migrace Návrh modifikace
6. Podpora uživatele	Přehled připomínek a požadavků uživatelů
7. Ukončení projektu	Ukončení projektu - Evidenční list

Tab. 12: Průběh kombinovaného projektu[167]

Fáze projektu	*Standardem požadované povinné dokumenty*
1. Zahájení projektu	Projektový záměr - Evidenční list
2. Zahájení koordinace subprojektů	Plán koordinace subprojektů Projektová bezpečnostní dokumentace IS

[166] Standard ISVS 005/02.01 pro náležitosti životního cyklu informačního systému [online]. Úřad pro veřejné informační systémy 2002. Dostupné na: http://www.mvcr.cz/micr/files/457/uvis_s005.02.01_v2002c5_20021218.pdf. str. 19 - 20

[167] Standard ISVS 005/02.01 pro náležitosti životního cyklu informačního systému [online]. Úřad pro veřejné informační systémy 2002. Dostupné na: http://www.mvcr.cz/micr/files/457/uvis_s005.02.01_v2002c5_20021218.pdf. str. 20 - 21

Fáze projektu	*Standardem požadované povinné dokumenty*
3. Monitorování subprojektů	Protokol o kvalifikačním testování systému Dokumentace subprojektů vyžadovaná standardem
4. Akceptace a ompletace	Protokol o převzetí IS
5. Ukončení projektu	Ukončení projektu – Evidenční list

15.3. Směrnice pro hodnocení a výběr CASE nástrojů, jmenovitě ČSN ISO/IEC 14102

Tato mezinárodní norma definuje jednak posloupnost procesů a jednak strukturovaný soubor charakteristik nástrojů CASE pro použití při technickém hodnocení a závěrečném výběru nástroje CASE. Vychází se z modelu hodnocení softwarového produktu popsaného v ISO/IEC 9126. Mezinárodní norma ISO/IEC 14102 přebírá obecný model charakteristik a subcharakteristik jakosti softwarového produktu popsaný v ISO/IEC 9126 a rozšiřuje jej na produkty typu nástrojů CASE: poskytuje charakteristiky jedinečné pro tyto nástroje. Tento obsáhlejší soubor charakteristik se pak organizuje do pěti skupin. Toto zformování do větších skupin zajišťuje dokonalejší přístup k řízení celého procesu hodnocení a výběru.

Tato mezinárodní norma poskytuje:

a) návod k identifikaci požadavků, které organizace má na nástroje CASE,

b) návod k mapování těchto požadavků do charakteristik, které mají být pro nástroj CASE hodnoceny,

c) proces pro výběr nejvhodnějšího nástroje CASE z několika nástrojů, který je založen na měření definovaných charakteristik.

Hlavními uživateli této mezinárodní normy jsou organizace, které mají v plánu převzít pro podporu procesů životního cyklu softwaru nástroje CASE. Tuto mezinárodní normu mohou používat také dodavatelé nástrojů CASE pro popis nástrojů.

Tato mezinárodní norma není určena pro použití pro prostředky softwarového inženýrství, jejichž účelem je poskytovat mechanismy pro datovou, řídicí a presentační integraci, pro nástroje všeobecného zaměření (např. textové procesory, tabulkové kalkulátory), které mohou být v softwarovém inženýrství používány, pro nástroje zaměřené na velmi úzkou oblast nebo na specifický účel (např. kompilátory) a pro plánování implementace nástrojů CASE v rámci organizace.

Technické hodnocení může odpovědět na otázku, jak dokonale nástroj CASE splňuje uživatelem stanovené požadavky. Cílem procesu technického hodnocení je poskytnout kvantitativní výsledky, na nichž může být založen konečný výběr nástroje. Závěrečný výsledek hodnocení by měl být objektivní, úplný a opakovatelný.

Tato mezinárodní norma pojednává o hodnocení a výběru nástrojů CASE a pokrývá celý životní cyklus softwarového inženýrství nebo jeho část. Zakládá procesy a činnosti, které se mají aplikovat pro hodnocení nástrojů CASE a pro výběr nejvhodnějších nástrojů CASE z několika kandidátů. Tyto procesy jsou generické a organizace je musí přizpůsobit svým potřebám. Procesy pro hodnocení a výběr nástrojů CASE by měly být zvažovány s ohledem na širší kontext s procesem technologie převzetí používaným v organizaci.

Uvedené informace pochází ze stránek Českého normalizačního institutu[168].

[168] Český normalizační institut, Dostupné na:
https://pdfonline.cni.cz/pdfonline/login.aspx,
http://pdfonline.cni.cz/html_nahledy/36/50263/50263_nahled.htm

16. Přehled použité literatury a zdrojů

Případy

, V. Analýza a návrh informačních systémů. Ekopress, Praha 1999, ISBN: 80-86119-13-0..19

Aldorf, F. Vývoj aplikací s použitím metodiky Rational Unified Process. Diplomová práce, VŠE 51

Ambler, S. W. More Process Patterns: Delivering Large-Scale Systems Using OO Technology, Cambridge University Press, 1999, ISBN 0-521-65262-6 50

Ambler, S. W. Process Patterns: Buiding Large-Scale Systems Using OO Technology, Cambridge University Press, 1998, ISBN 0-521-64568-9.....50

Arlow, J. UML a unifikovaný proces vývoje aplikací. Computer Press, Brno 2003, ISBN 80-7226-947-X 43

Banker, R.D. Kaufmann, R.J. Reuse and Productivity in Integrated Computer-aided Software Engineering: An Emperical Study, MIS Quaterly, 15:3, 1991 109

Beck, K. Extrémní programování. Grada, Praha 2002, ISBN. ISBN 80-247-0300-9.....57

Beneš, M. Přehled OO metodik a notací. Diplomová práce. VŠE 2005. kap. 3.3: Srovnání možností vybraných programů pro modelování 133

Boehm, B. Egyed, A. Kwan, J. Port, D. Shah, A. Madachy, R. Using the WinWin Spiral Model: A Case Study, červenec 1998.....47

Buchalcevová, A. Metodika feature-driven development neopouští modelování a procesy, a přesto přináší výhody agilního vývoje. sborník konference Tvorba softwaru. Ostrava 2005.....59

Buchalcevová, A. Metodiky budování IS/ICTNávrh metodického rámce IS/ICT: Metodiky budování IS/ICT. KIT VŠE Praha 2004 48

Buchalcevová, A. Metodiky vývoje a údržby informačních systémů. Grada, Praha 2005, ISBN 80-247-1075-7 .....passim

Buchalcevová, A. Návrh metodického rámce IS/ICT: Metodiky budování IS/ICT. KIT VŠE 2004 .....passim

Buchalcevová, A. Pavlíčková, J. Pavlíček, L. Základy softwarového inženýrství Oeconomica, Praha 2007, 1. vyd. ISBN 978-80-245-1270-9 15

Buchalcevová, A. Stav používání agilních metodik v ČR. Systémová integrace 2006, cis. 4 ,ISSN 1801-1578.....55

Buchalcevová, A.: Metodiky vývoje programových systémů, In: sborník prací účastníků vědeckého semináře doktorského studia FIS VŠE v Praze, 2003, ISBN 80-245-0518-5....33

Capability Maturity Model V1.1 Pocket Guide. Judy Bamberger. Merant, ltd.. 1999 ..49

Copeland, L. Extreme Programming, Computerworld 2001, ISSN 0010-4841 55

Felsing J, Palmer, M. S. R. A Practical Guide to Feature - Driven Development. Prentice Hall 2002. ISBN 0130676152.....59

Finlay, P.N. Mitchell, A.C. Perceptions of the Benefits from the Introduction of CASE: An Emperical Study, MIS Quaterlym 18:4, 199.....109

Fuggetta, A. A Classification of CASE technology. Computer, December 1993.....109

Houser, P. Computer Aided Software Engineering (CASE). Business World, únor 2007, ISSN: 1682-3257 61

Hradecká, P. Knowledge management. Diplomová práce, VŠE Praha 2006 83

Kadlec, V. Agilní programování: Metodiky efektivního vývoje softwaru. Computer Press, Brno 2004 57

Kadlec, V. Agilní programování: Metodiky efektivního vývoje softwaru. Computer Press, Brno 2004, 1. vyd. ISBN: 80-251-0342-0 ..... passim

Kendall, K.E. Systém Analysis and Design. Prentice Hall, 1991 26

Lending, D. Chervany, N. L. Case tool use and job design: A restrictivness/flexibility explanation, The Journal of Computer Information System, Fall 2002 125

Likelihood methods for testing group problem solving models with censored data. Psychometrika 1978, vol. 43 39

McMurtrey, M. E. Teng, J. T. C. Grover, V. Hemant, V. Current utilization of CASE technology. Industrial Management & Data Systems 2000, roc. 100, cisl 1. str. 22 - 30, ISSN 0263-5577 70

Pergl, R. Analýza vnitřních vazeb principů metody extrémního programování. Sborník .57

Pergl, R. Struska, Z. Čím mohou přispět nejznámější agilní metodiky ke zlepšení vývojového procesu. ČZU PEF Praha 2002 ..... 57, 58, 59

Polák, J. Merunka, V. Carda, A. Umění systémového návrhu. Grada, Praha 2003, ISBN: 80-247-0424-02 26

Poppendieck, M. Poppendieck, T. Lean Software Development: An Agile Toolkit for Software Development Managers. Addison-Wesley 2003. ISBN 0321150783 60

Pressman, R. S. A Pracittioner´s Approach: Software Engineering. McGraw-Hill, New York, 1992 48

Ringoš, J. Extrémní programování. PC Svět, srpen 2004 , ISSN 1213-6042 55

Rubin, H. Worldwide Benchmark Project Report. Rubin Systems Inc. 1995 62

Řepa, V. Analýza a návrh informačních systémů. Ekopress, Praha 1999, ISBN: 80-86119-13-0 ..... passim

Řepa, V. Chlapek, D. Materiály ke strukturované analýze. Vysoká škola ekonomická, Praha 1997, ISBN 80-7079-260-4 ..... 19, 22, 61, 65

Schwaber, K. Sutherland, J. Scrum Development Process. OOPSLA Business Object Design and Implementation Workshop 1997. Springer: London 58

Smil, P. Úvod do tvorby IS s použitím objektového přístupu. Softwarové noviny, prosinec 1999, cis.12, ISSN: 1210-8472 ..... 34, 76

Sodomka, P. Investujte a vydělejte. BIZ, Computer Press Brno 2007, ISSN ISSN: 1213-063X 124

Synek, M. Manažerská ekonomika. Grada Publishing, Praha 2000, 4. vyd. ISBN 80-. 247-9069-9 126

Takeuchi, H. and I. Nonaka. The New New Product Development Game. Harvard Business Review 1986 57

Technologie - rychlokurz: Iterativní vývoj softwaru, Computerword srpen 2007, ISSN: 1210-9924 43

Vodáček, L. Rosický, A. Pojetí, poslání a význam informačního managementu. Management Press, Praha 1997, ISBN 80-85943-35-2 35

Voříšek, J. Strategické řízení informačního systému a systémová integrace. Management Press, Praha 1999, ISBN: 80-85943-40-9 ..... 19, 69

Wiley, J. Putting the Software Engineering into CASE by K. Robinson, published by John Wiley and Sons Inc. New York 1992 64

Statuty

Ambler, S. W. Enterprise Unified Process (EUP) [online]. Agile Strategies for Enterprise IT. Ambysoft Inc. 2007 Dostupné na: http://www.enterpriseunifiedprocess.com 52

Analýza IS, modely životního cyklu IS [online]. Dostupné na: http://iss.unas.cz/ids/02.pdf 45

Balogh, I. Jankó, G. Murín, J. Žilka, R. Nástroje meta-CASE (charakteristika, prehled trhu, trendy) [online]. VŠE 2005. Dostupné na http://panrepa.org/CASE/meta_CASE.pdf .77

Beck, K. Beedle, M. Bennekum, A. Cocburn, A. Cunningham, W. Fowler, M. Greening, J. Highsmith, J. Hunt, A. Jeffries, R. Kern, J. Marick, B. Martin, R. C. Mellor, S. Schwaber, K. Sutherland, J. Thomas, D. Manifesto for Agile Software Development [online]. 2001., Dostupné na: http://agilemanifesto.org/ 54

Bézivin J. Who´s Afraid of Ontologies [online]. Proceedings of OOPSLA´98 Workshop No.25 – CDIF. Vancouver 1998. Dostupné na: http://www.metamodel.com/oopsla98-cdif-workshop/bezivin1 75

Bézivin, J. Who is Afraid of Ontologies [online]. Proceedings of OOPSLA´98 Workshop No.25 – CDIF, Vancouver 1998. Dostupné na: http://www.metamodel.com/oopsla98-cdif-workshop/bezivin1 75

Bitpipe: The TechTarget Library of White Papers, Product LIterature, Webcasts and Case Studies [online]. Dostupné na: http://www.bitpipe.com/tlist/Computer-Aided-Software-Engineering.html 61

Boehm, B. A Spiral Model of Software Development and Enhancement [online]. Computer May 1988. Dostupné na: http://www.ieee.org/portal/site 44

Buchalcevová, A. Agilní a rigorózní metodiky: Úrovně zralosti CMM. Dostupné na: http://nb.vse.cz/~vorisek/FILES/4IT215_materialy_k_predmetu/MetodikyIT_Buchalcevova.ppt 49

Carnegie Mellon: Software Engineering Institute. What is a CASE Environent [online]. Carnegie Mellon University 2007. Dostupné na: http://www.sei.cmu.edu/legacy/case/case_whatis.htm 61

Co jsou agilní metodiky vývoje software [online]. Reengine 2008. Dostupné na: http://www.reengine.cz/index/agilni_metodiky.do 54

Čáp, J. Obrázek, M. Růžek, P. Turek, J. Smetana, J. Přehled nástrojů CASE na tuzemském trhu [online]. VŠE 2008. Dostupné na: http://panrepa.org/CASE/jaro2008/case_jaro2008.pdf 102

Český normalizační institut, Dostupné na: https://pdfonline.cni.cz/pdfonline/login.aspx, http://pdfonline.cni.cz/html_nahledy/36/50263/50263_nahled.htm 144

Dubravec, P. Case nástroje [online]. Dostupné na: http://www.dubravec.cz/dubravcovi/cl000001.htm#a8 68

Dudka, K. Softwarové inženýrství [online]. Dostupné na: http://dudka.cz/studyIUS 15

Fuggetta, A. Milano, P. CERFIEL A Classification of CASE Technology [online].The world's leading professional association for the advancement of technology 1993. Dostupné na: http://www.ieee.org/portal/site 79

Fuggetta, A. Milano, P. CERFIEL A Classification of CASE Technology [online].The world's leading professional association for the advancement of technology 1993. Dostupné na: http://www.ieee.org/portal/site (navštíveno 5. dubna 2008) 48

Hayes, S. Andrews, M. An Introduction to Agile Methods [online]. Wry Tradesman 2004. Dostupné na: http://www.wrytradesman.com/articles/IntroToAgileMethods.pdf.....55

Hovězák, V. Historie počítačů. Dostupné na: http://www.kyberman.wz.cz/files/1_Historie_pocitacu.pdf.....15

Hřebejk, P. Řepa, V. Příspěvek na konferenci Data-Sem 99. Meta-modelování. praha 1999. Dostupné na: http://nb.vse.cz/~repa/veda/RaD.htm.....75

IEEE Standard Glossary of Software Engineering Terminology: IEEE std 610.12-1990 [online]. IEEE 2004. Dostupné na: http://standards.ieee.org/reading/ieee/std_public/description/se/610.12-1990_desc.html.....14

Informační systémy [online]. Unicorn 2008. Dostupné na: http://www.unicorn.eu/cz/produkty/systems/index.php?id=15630.....43

ISO/IEC 12207 Software Life Cycle Processes [online]. SEPT Supplying Software Engineering Standards Information to the World 2008. Dostupné na: http://www.12207.com.....29

Jánský, V. Kříž, P. Šebelík, J. Podpora plánování a řízení projektů v CASE nástrojích [online]. prosinec 2005. Dostupné na: http: //panrepa.org/CASE/CASE_vs_RIP.pdf...64, 69

Kadlec, V. Case a zbytek světa II., požadavky [online]. Databázový svět 2002. Dostupné na: http://www.dbsvet.cz/view.php?cisloclanku=2002062801.....63

Kadlec, V. Case a zbytek světa II., požadavky [online]. Databázový svět 2002. Dostupné na: http://www.dbsvet.cz/view.php?cisloclanku=2002070104.....63

Kadlec, V. Vyvíjíme databázovou aplikaci: 3. díl jak na to [online]. Databázový svět 2003. Dostupné na: http://www.dbsvet.cz/view.php?cisloclanku=2003051203.....46

Kaluža, R. Case nástroje [online]. radovan.bloger.cz 2006. Dostupné na: http://radovan.bloger.cz/it/informacni-systemy/case-nastroje.....66

Kick-off meeting: Základ dobré organizace implementace [online]. essence business solutions 2008. Dostupné na: http://www.essencebs.cz/cz/jak-vedeme-projekty/2-kick-off-meeting.....128

Klobasa, P. Obhajoba vodopádového vývoje [online]. Oxyonline: vývojářský blog 2008. Dostupné na: http://vyvojari.oxyonline.cz/vodopadovy-vyvoj-obhajoba.....41

Kraval, I. Návrh informačních systémů pomocí UML, OOP a vzorů [online]. Server objektových technologií 2006. Dostupné na: http://www.objects.cz/clanky/clanky_IS/NavrhIS.pdf.....37, 38, 39

Lifecycle Models. National Instruments [online]. Dostupné na: http://zone.ni.com/reference/en-XX/help/371361A-01/lvdevconcepts/lifecycle_models.....36

Meecham, B. CASE [online]. Midmarket CIO Definitions 2005. Dostupné na: http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci213838,00.html#.....61

Merunka, V. Metody tvorby informačních systémů [online]. PEF ČZU Praha 2008. Dostupné na: http://kii.pef.czu.cz/~merunka/documents/for_students/U3V/U3V%20-%20metody%20tvorby%20IS%20-%20text.doc.....35

Miller, G. A. Fellbaum, Ch. Tengi, R. Wakefield, P. Langone, H. Haskell B. R. WordNet: lexical database [online]. Princeton University. Dostupné na: http://wordnet.princeton.edu.....18, 21

Modell, M. E. A Professional's Guide to Systems Analysis: Glossary of Terms and Concepts [online]. McGraw-HillBook Company New York 2007. Dostupné na: http://www.martymodell.com/pgsa2/pgsa-glossary.html 65

Molhanec, M. Úvod do datového modelování [online]. ČVUT FEL 2008. Dostupné na: http://martin.feld.cvut.cz/~mmm/Vyuka/X13DFA/files/UdDM.pdf 66, 67

Nešetřil, V. Fázová organizace projektu a průběhové modely [online]. Vysoká škola báňská - Technická univerzita Ostrava 2001. Dostupné na: http://fmmi10.vsb.cz/639/qmag/mj19-cz.htm#k 41

Object Management Group. Meta Object Facility [online]. Dostupné na: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=32622 77

Panter, S. Software Best Practice (ESSI) [online]. ESSI 1997. Dostupné na: http://cordis.europa.eu/esprit/src/essi.htm 62

Pavelka, J. Softwarové inženýrství [online]. Matfyz 2008. Dostupné na: http://wiki.matfyz.cz/wiki/SWI026_Pavelka#Model_vodop.C3.A1d 44

Procházka, J. Nástroje CASE [online]. Databázový svět 2004. Dostupné na: http://www.dbsvet.cz/view.php?cisloclanku=2004052702 71

Pstružina, K. Atlas filosofie vědy (Fond rozvoje MŠMT F5 1588/2002) [online]. Dostupné na: http://nb.vse.cz/kfil/win/atlas1/atlas3.htm 18

Řepa, V. Metodika vývoje informačního systému s pomocí nástroje Power Designer [online]. Sybase ČR 2006. Dostupné na: http://www.sybase.cz/buxus/docs/Metodika_vyvoje_IS_06_2006.pdf 23

Scrum Concept [online]. ScrumAlliance. Dostupné na: http://www.scrumalliance.org/view/scrum_concept 58

SO/IEC 15504 Software Process Improvement and Capability Determination. Information technology Process assessment Part 1: Concepts and vocabulary, Information technology Process assessment Part 2: Performing an Assessment, Information technology Process assessment Part 3: Guidance on performing an assessment, Information technology Process assessment Part 4: Guidance on use for process improvement and process capability determination, Information technology Process Assessment Part 5: An exemplar Process Assessment Model, Information technology Process assessment Part 6: An exemplar system life cycle Process Assessment Model, Information technology Process assessment Part 7: Assessment of Organizational Maturity [online], Dostupné na: http://www.iso.org/iso/iso_catalogue.htm 29

Standard ISVS 005/02.01 pro náležitosti životního cyklu informačního systému [online]. Úřad pro veřejné informační systémy 2002. Dostupné na: http://www.mvcr.cz/micr/files/457/uvis_s005.02.01_v2002c5_20021218.pdf ..... passim

Šenovský, P. Case systémy [online]. VŠB-TUO FBI 2006. Dostupné na: http://homel.vsb.cz/~sen76/case/2CASE.doc 65

Terminologie. Česká společnost pro systémovou integraci [online]., Dostupné na: http://www.cssi.cz/cssi/terminologie 31

The Yourdon (Ward-Mellor) Structured Method [online]. Schools of Electronic Engineering and Computer Science. Dostupn0 na: http://www.informatics.bangor.ac.uk/~dewi/modules/rts/YSM_slides.pdf 36

Vlk, T. Manifest agilního vývoje softwaru z osobní perspektivy [online]. Tril 2008. Dostupné na http://www.tril.cz/manif.html 54

Wikipedie: Otevřená encyklopedie [online]. Dostupné na: http://cs.wikipedia.org/wiki/Metodologie 19

Zákon o veřejných zakázkách č. 137/2006 Sb. [online]. 2008. Dostupné na: http://www.verejna-zakazka.cz/zakon.....120

Pravidla

Seznam obrázků

Seznam tabulek

17. Terminologický slovník

Následující tabulka obsahuje vysvětlení termínů, které se nacházejí v této diplomové práci. Termíny pochází z terminologického slovníku Katedry informačních technologií.

Termín	Význam
B2B	Business-to-Business Obchodní vztahy se elektronicky realizují mezi dvěma podniky, resp. právními subjekty, na bázi výměny strukturovaných dat (objednávky, potvrzení, faktury,..). viz EDI, XML. Zahrnuje všechny komerční transakce mezi dvěma firmami, které jsou prováděny pomocí elektronických prostředků". Významným rysem modelu B2B je větší důraz na logistiku a zajištění samotného obchodu, oproti důrazu na získání zákazníka, jako je tomu v případě obchodů B2C.
B2C	Business-to-Customer Obchodní vztahy mezi podnikem a konečným spotřebitelem, realizované webovými aplikacemi, virtuálními obchody na Internetu apod. On-line obchodování na Internetu, tj. prodej zboží (ať už hmotného či nehmotného) a služeb koncovým zákazníkům pomocí služby World Wide Web. Tato oblast je označována i jako „ecommerce". Oblasti B2C můžeme rozdělit do tří oblastí: ▪ Prodej informací – tzv. „bit business". Zde je možné produkt kompletně distribuovat elektronickou cestou. do této skupiny patří prodej a pronájem softwaru nebo např. publikování informací (elektronické noviny, burzovní zprávy, hudební servery apod.). ▪ Prodej zboží – produkt je objednán a případně i zaplacen elektronicky, jedná se však o hmotné zboží. ▪ Poskytování reklamního prostoru – podmínkou je vlastnictví dostatečně navštěvovaného serveru.
B2G	Business-to-Government Vztahy mezi podnikem a státní správou (finančními úřady, pojištěním, orgány místní správy, …), většinou na bázi výměny strukturovaných dat.
BI	Business Intelligence BI je datový systém pro podporu rozhodování. Podle Power, J.D., BI popisuje jako sadu konceptů a metod vycházejících z faktů na základě reportů a dotazovacích nástrojů. BI nabízí historická a současná data včetně prediktivních údajů získaných na základě trendů dostupných dat.

Termín	Význam
BPMN	Business Process Modeling Notation BPMN - diagram, grafický způsob znázornění událostí, aktivit a jejich vzájemnou kauzalitu. Jako standard je prosazován OMG. Více na http://www.bpmn.org/
CASE	Computer Aided System Engineering (možno se také setkat s názvem Compuetr Aided Software Engineering) - modelování systémů pomocí počítačových nástrojů, jejichž podporu lze využít v celém životním cyklu vývoj software / modelu systému
CASE nástroj	Počítačový program používaný při modelování systémů (viz CASE), který je schopen grafické reprezentace modelů nebo diagramů systému a prvků těchto systémů. Často jsou tyto nástroje vybaveny dalšími funkcemi pro korekci správnosti modelů.
CORBA	Common Object Request Broker Architecture CORBA je objektový model pro distribuované aplikace nezávislý na platformě, vyvíjený sdružením OMG. Základním znakem standardu je nezávislost na implementačním jazyku, komunikačním protokolu nebo inforačním systému.
CRM	Customer Relationship Management CRM je systém prpo podporu řízení vztahů se zákazníkem, který shromažďuje data o chování zákazníků, kontaktní informace a další data za účelem poskytování lepších služeb.
databáze	1) Datová základna je integrovaná počítačově zpracovávaná množina dat; 2) Data, která jsou využívána ve více aplikacích. V databázi jsou minimalizovány redundance dat a existuje vhodně organizovaná správa těchto dat. Cílem databázového systému je uspořádat datové zdroje (datovou základnu) na počítači tak, aby tyto zdroje mohly být využívány více uživateli a mohly být využity na různých počítačích zapojených do sítě. Základní komponentou databázové koncepce je programový systém umožňující práci s databází – SŘBD (Systém řízení báze dat, angl. DBMS – Data Base Management System); 3) Systém řízení báze dat je souhrn programových prostředků, který umožňuje: ▪ vytvoření databáze a definici její struktury (dle příslušného modelu dat – viz dále), ▪ manipulaci s uloženými daty (operace Insert, Update, Select, Delete) a sdílení uložených dat více aplikacemi a uživateli, správu databáze. Hlavní přínosy databázového přístupu jsou: ▪ sdílení dat,

Termín	Význam
	▪ snížení redundance dat, ▪ snazší zabránění vzniku nekonzistencí, ▪ podpora transakčního zpracování, ▪ údržba integrity databáze, ▪ zajištění ochrany databáze před neautorizovaným přístupem. Modely dat, na jejichž principech jsou vytvářeny jednotlivé databázové systémy: ▪ Hierarchický model dat (např. IMS, Systém M), ▪ Síťový model dat (např. IDS, IDMS), ▪ Relační model dat a v současné době i relačně objektový model dat (např. Oracle, Informix, Sybase, MS SQL Server, Progress, DB/2), ▪ Objektový model dat (např. Orion, GemStone, Ontos, Object Store). 4) Databázový systém je souhrnné označení pro systém řízení báze dat a datovou základnu jím spravovanou.
DBMS	Data Base Management Systém DBMS je skupina programů, které fungují jako rozhraní mezi databází a uživatelem. Slouží pro manipulaci s databází a daty v ní.
DFD	Data Flow Diagram Data Flow Diagram neboli diagram datových toků je diagram, který zobrazuje toky dat mezi procesy, soubory a externími prvky v modelovaném IS.
ERP	Enterprise resource planning ERP je manažerský systém, který integruje a automatizuje velké množství procesů souvisejicich s produkčními činnostmi podniku.
ICT	Informační a komunikační technologie Hardwarové a softwarové prostředky pro sběr, přenos, ukládání, zpracování a distribuci dat. Mezi hardwarové (technické) prostředky patří - servery, stacionární a přenosné personální počítače, tiskárny, komunikační a síťová zařízení (vysílače, směrovače, přepínače) a specializovaná koncová zařízení (myš, tablet, scanner, kamera, PDA, mobilní telefon apod.). Mezi softwarové (programové) prostředky patří základní software (operační systém, databázový systém, komunikační systém), aplikační software a software pro modelování a vývoj informačních systémů.
IEEE	Institute of Electrical and Electronics Engineers IEEE je mezinárodní nezisková organizace s rozsáhlou publikační činností, spravující technické standardy, rejstříky a dokumentace.

Termín	Význam
	Více na www.ieee.org
implementace informačního systému	1) Jedna z etap tvorby informačního systému - Životní cyklus vývoje informačního systému. Hlavní činností při implementaci je programování spolu s doprovodnými činnostmi (testování, ověřování správnosti). 2) v oblasti TASW: Činnosti od pořízení TASW po předání TASW do provozu (Analýza, návrh IS, nastavení parametrů a úpravy chování TASW pro konkrétní podmínky (customizace), vytvoření dokumentace, vyškolení uživatelů).
informace	Informace je význam, který člověk v procesu interpretace přisuzuje datům. Hlubší porozumění pojmu „informace" vyžaduje překonat běžné pojetí, které se váže na informaci presentovanou člověkem pomocí (jazykových) symbolů. Ty mohou být prezentovány i pomocí informačních a komunikačních technologií ICT. Mimo informace presentované pomocí symbolů přijímá člověk z okolí i jiné formy informace (které nejsou vyjadřovány pomocí symbolů) a které ovlivňují jeho kognitivní procesy včetně tvorby znalostí a iniciovaní aktivit. Obecnému pojetí informace věnuje značnou pozornost moderní informační věda.
informační systém	Informační systém je systém jehož prvky jsou informační a komunikační technologie, data a lidé. Cílem informačního systému je efektivní podpora informačních a rozhodovacích procesů na všech úrovních řízení organizace (podniku). Při návrhu a implementaci nové verze informačního systému organizace je třeba brát v úvahu řadu aspektů (dimenzí) a jejich vazeb: ▪ hardware - z jakých technických komponent bude vytvořena technologická infrastruktura informačního systému, ▪ software - jaké softwarové komponenty bude informační systém obsahovat a jaká bude funkcionalita a vzájemné vztahy těchto komponent, ▪ data - jaká data budou v informačním systému uložena, jaká data budou z okolí (např. od dodavatelů, od zákazníků) do IS vstupovat, jaká data naopak budou z IS do okolí poskytována, ▪ procesy - na podporu kterých podnikových procesů a činností bude IS sloužit a jakým způsobem, ▪ informační služby - služby, které bude IS poskytovat svým uživatelům, Služba je vymezena zejména dodavatelem služby, uživatelem služby, funkcionalitou, objemem (počet uživatelů, objem zpracovávaných dat), kvalitou (dostupnost, doba odezvy a zabezpečení jednotlivých aplikací) a cenou, ▪ lidé - kdo bude uživatelem jednotlivých aplikací/služeb a jejich funkcionality a jaké jsou kvalifikační předpoklady pro efektivní využití aplikací/služeb, ▪ organizace - jaké organizační změny a změny předpisů si

Termín	Význam
	implementace nové verze IS vyžádá, ▪ legislativa - jaké zákony a normy se k provozování jednotlivých aplikací vztahují, ▪ ekonomika - jaké náklady a jaké přínosy jsou s provozem IS spojeny. Informační systém - dle normy ČSN ISO 5127-1 jde o "komplex lidí, informací, systému řízení chodu IS, který zabezpečuje těsné a logické propojení na prostředí, systém organizace práce spojený s provozem a využitím IS, technické prostředky a metody zabezpečující sběr, přenos, aktualizaci , uchování a další zpracování dat za účelem tvorby a prezentace informací pro potřeby uživatelů a použité informační. technologie."
internet	Globální celosvětová počítačová síť propojující regionální a rozsáhlé počítačové sítě, které používají TCP/IP jako síťový protokol. Množina -> komunikačních (počítačových) sítí, které jsou vzájemně propojeny na základě bilaterálních nebo multilaterálních smluv a vytvářející globální (celosvětovou) síť. V rámci této množiny mohou oprávnění uživatelé využívat jak přenosových kapacit sítí, tak zdrojů, které jsou do sítí připojené (počítače, servery, služby, data). Jednotlivé prvky sítí při komunikaci využívají dohodnutá pravidla, která jsou označována jako protokoly v architektuře -> TCP/IP.
ISVS	Informační Systémy Veřejné Správy - soubor informačních prostředků použitých pro výměnu informací a přístup k datovým strukturám v rámci veřejné správy, jehož správcem je MVČR. Více na http://www.sluzby-isvs.cz/
J2EE	Java 2 Enterprise Edition - metodika, jak provádět design, vývoj, nasazení a provozování vícevrstvých aplikací v jazyce Java (viz Java)
Java	Objektově orientovaný na platformě nezávislý programovací jazyk vyvíjený společností SUN Microsystems.
knowledge management	Řízení znalostí (KM) Knowledge management je princip řízení, který považuje znalosti za aktivum organizace, které jí přináší trvalou konkurenční výhodu. Je to manažerská disciplína, která podporuje integrovaný přístup k identifikaci, hodnocení, zachycení, tvorbě, analýze, sdílení a použití intelektuálních zdrojů společnosti. Více viz. řízení znalostí.
komunikace	Předávání údajů mezi objekty, může to být předání zprávy nebo volání metody.
outsourcing	

Termín	Význam
	V řízení podniku: Strategický organizační nástroj. Přesun odpovědnosti za oblast činností podniku na externí specializovanou firmu - poskytovatele; zpravidla včetně zaměstnanců a vlastnictví aktiv; především za účelem zaměření na hlavní činnost, dosažení náležité úrovně kvality v oblasti, případně úspory nákladů. Aplikuje se u oblastí, které nejsou hlavními činnostmi podniku, tedy nejsou motorem dlouhodobé konkurenceschopnosti podniku. Outsourcing se používá i v oblasti podnikových informačních systémů. Úroveň služeb poskytovaných formou outsourcingu se sjednává ve formě SLA.
Merise	Francouzská metodika systémového přístupu k analýze a tvorbě informačních systémů.
OMG	Object Management Group - mezinárodní nezisková počítačová ogranizace, zabývající se definováním standardů pro rozsáhlou řadu technologií, spravující standard UML, SYSML, BPMN, OMG MDA, DDS, CORBA, MOF a další. Více na www.omg.org
OMG MDA	OMG Model Driven Architecture - Architektura pro vývoj software, navržená OMG, definující modelové apliakce intergující bussiness chování a funkcionalitu. Více na www.omg.org
OMG MOF	OMG MetaObject Facility - souhran modelových standardů, díky kterým mohou být modely volně přenášeny mezi aplikacemi, včetně přenosu po síti, uložení v repository, vyexportovány do jiných formátů. Převzato z www.omg.org. Více na www.omg.org/mof/
outsourcing IS/ICT	Podstatou outsourcingu IS/IT je zajišťování vybraných činností a služeb IS/IT externími dodavateli. Důvody pro toto řešení mohou být konkurenční, odborné, finanční nebo organizační. Podle toho, co je předmětem outsourcingu se rozlišuje outsourcing rozvoje IS/IT, tj. implementace jednotlivých standardních aplikací a technologií, případně vývoj specializovaných aplikací přímo podle potřeb podniku. Vedle toho je další možností i outsourcing provozu IS/IT, tj. provozování jednotlivých aplikací, případně celého systému na technice a SW dodavatele, případně zákazníka, avšak s tím, že se dodavatel stará i o údržbu a inovaci této „zapůjčené“ techniky. Totální outsourcing pak znamená, že dodavatel zajišťuje provoz a rozvoj zákazníkovi kompletně.
proces	1) ve smyslu podnikového procesu se procesem rozumí řízená posloupnost činností, spojená definovaným účelem, s cílem vyprodukovat definovaný výstup (produkt, službu). Řízení podnikového procesu je založeno na poznání objektivních zákonitostí a souvislostí typových dějů v rámci oboru činnosti podniku a jejich formalizaci v podobě základního předpisu procesu a pravidel pro jeho řízení.

Termín	Význam
	2) v oblasti informačních systémů: Reakce systému na určitou událost nebo kombinaci událostí. Reakce je reprezentována sledem činností a funkcí IS. Vykonání určité činnosti, resp. funkce IS může být vázáno na určitou podmínku. Tento pojem úzce souvisí s pojmem -> Funkce IS, neboť při detailnějším pohledu můžeme zkoumat, jakým Procesem je daná Funkce zajišťována. Pojem Proces tedy akcentuje posloupnost činností (operací), kterou je daná funkce realizována, kdežto pojem Funkce akcentuje vstupy a výstupy funkce jako celku. 3) ve výpočetní technice: Program spuštěný a probíhající.
proces	1) Předpis posloupnosti činností pro podřízené či výkonné elementy (program zájezdu, televizní program, program rozvoje firmy). 2) Program ve výpočetní technice – předpis činnosti počítače. ¨Velké programy = programové systémy, např. operační systém, databázový systém. Počítačový program je posloupnost příkazů proprocesor počítače. Příkazy jsou jednak sekvenční (po provedení jednoho se provede následující), jednak rozhodovací, kdy se volí jedna ze dvou větví sekvenčních posloupností příkazů. Program je reprezentován souborem v paměti počítače (obvykle s příponou „exe“ nebo „com“), součástí programu však může být množství dalších souborů nejrůznějších typů. K zápisu programu v souboru (textovém) je použit programovací jazyk, tomuto souboru se říká zdrojový program. Základní dělení programů: interaktivní (střídají se činnosti člověka a počítače řízeného programem), dávkový (člověk nejprve připraví všechna data a program pak pracuje bez zásahu člověka) nebo systémový (nekomunikuje s člověkem ale s jinými programy nebo technickými zařízeními).
RUP	Rational Unified Process - metodika používaná pro vývoj systémů, která obsahuje celý životní cyklus vývoje softwaru. Je vytvořená společností Rational Software Corporation a je používaná v jejích produktech.
řízení znalostí	1) Systematický proces hledání, vybírání, organizování, destilování a prezentování informací způsobem, který zlepšuje porozumění pracovníka specifické oblasti zájmu. [Thomas H. Davenport] 2) Proces „chytání“ kolektivní podnikové odbornosti.[Justin Hibbard] 3) Formulace podnikové strategie pro rozvoj a aplikaci znalostí, které přispějí ke zlepšení podnikových procesů a schopnosti reakce.[Owen Wilson] 4) Procesy plánování, zajišťování, vytváření a využívání znalostí, které účelně a účinně zabezpečují potřeby myšlenkových procesů i jednání lidí. [Leo Vodáček] 5) Ta část managementu podniku, která řídí proces vytváření, aktualizace a aplikace intelektuálního kapitálu s cílem dosažení

Termín	Význam
	prosperity. [Pavel Čech]
SEO	Search Engine Optimization - metoda upravení internetové stráýnky tak, aby u indexovaxíxh robotů stránky získaly co nejvyšší rating a umístily se co nejvýše ve vyhledávacích portálech, což přiláká na stránky více návštěvníků - potencionálních zákazníků.
server	1) Samostatný program, který je trvale spuštěn a který očekává požadavky jiného programu (klienta). Úkolem serveru je vedle přijímání požadavku, vykonání činností, které jsou mu naprogramovány, a formulace odpovědi programu, který požadavek vznesl; 2) (přeneseně) technický prostředek (počítač), na kterém je spuštěn proces serveru.
softwarová krize	Pojmenována podle situace v 60. letech, začala se projevovat neúnosným prodlužováním a prodražováním projektů, nízkou kvalitou výsledných produktů, problematickou údržbou a nízkou produktivitou práce programátorů. Částečně vyřešena zavedením strukturovaného programování.
SŘBD	Systém řízení báze dat, viz DBMS.
SSADM	Structured Systems Analysis and Design Method - standard systémového přístupu k analýze a tvorbě informačních systémů, vyvinutý pro CCTA, státní organizaci ve Velké británii, zabývající se použitím IT ve státní správě.
SW	Software Programy, procedury a pravidla pro zpracování konkrétní úlohy na počítači neboli pokyny počítači, jak má danou úlohu řešit. Program je napsán v programovacím jazyku (např. Java, C, Pascal, Cobol, assembler). Software se dělí na základní (operační systém, databázový systém, komunikační systém) a aplikační.
systém	Systém je celkovostní entitou vyjadřující vnitřní vazby (vztahy), organizaci a funkce jednotlivých prvků této entity. Systém může být materiální nebo abstraktní.
UML	Unified Modelling Language Tento jazk představuje základ Model-Driven Architecture, což je standardizace společnosti OMG, která sjednocuje kroky během vývoje a integrace systémů. Jazyk obsahuje podporu pro … aktuální verze jazyka je 2.0, ktrá jeji rošiřuje oproti verzi 1.0 zejména o specifikaci superstruktur.
vývoj IS	Proces a vědomá činnost směřující k dosažení předem určených cílů -> informačního systému (definovaných v Informační strategii).

Termín	Význam
	Vývoj IS probíhá většinou prostřednictvím sady projektů IS/ICT rámcově definovaných v Informační strategii. Vývoj IS zahrnuje základní skupiny procesů/činností: specifikaci požadavků, koncepci, analýzu, návrh (design), implementaci, zavedení a provoz/údržbu (konkrétní rozdělení do etap a činností definují metodiky vývoje IS).
WorkFlow Diagram	Obecný diagram graficky reprezentující popis procesů nebo jednotlivých kroků v procesu a jejich posloupnosti.
XML	eXtensible Markup Language XML je standard vyvinutý konsorciem W3C, používaný při přenášení dat a publikování dokumentů. Standard XML umožňuje vytvoření libovolného souboru dat s podporou irokého spektra datových typů.
znalost	1) Informace v kontextu, umožňujícím jejich využití při rozhodování. Kontext je zde vytvářen především zkušenostmi/schopnostmi a může mít podobu metodik, technik, formálních postupů apod. Převažuje rozlišení dvou typů znalosti [Nonaka, Takeuchi]: a) Formální, explicitní (Explicit) má hmotnou podobu, zpravidla jako data v informačním systému (IS) podniku. Tato znalost je obsažena především v podnikových procesech, definujících kontext užití informací z IS, b) Neformální, skrytá (Tacit) - osobní, neformální, často nevědomá, obtížně sdělitelná. Má podobu osobních ideálů, postojů, intuice, tušení, sympatií atp. 2) Dynamicky fungující systém vzájemné interakce osvojených zkušeností, získaných poznatků a člověkem sdílených hodnot i názorů v jeho myšlenkových procesech. Tvoří systémový rámec pro interpretaci a transformaci dat na informace, jejich vyhodnocování a integraci do již existujících znalostí [Vodáček]. Charakteristické rysy znalostí jako pojmu jsou [Svátek, Berka, Pstružina, Truneček, Vejlupek, Zdráhal]: ▪ jsou obecné (oproti datům, která jsou specifická), ▪ nejsou objektivně verifikovatelné (oproti datům, která jsou), ▪ jsou individuální, oproti "vědění", které je kolektivní. Existují však i znalosti vlastněné skupinou lidí (např. „znalostní kapitál“, či „kultura“ firmy), ▪ jsou formalizovatelné (oproti dovednostem či schopnostem), jakkoliv nemusejí být formalizované (viz tzv. "tacitní" znalosti diskutované výše), ▪ jsou vzájemně provázané (oproti informacím či datům, které jsou izolované, týkají se dílčích jevů), ▪ týkají se procesu (oproti informacím, které jsou statické).
životní cyklus	Přístup, který procesně popisuje vývoj IS. Jeho cílem je realizace

Termín	Význam
vývoje informačního systému	a zprovoznění nové verze IS. Je tvořen několika fázemi: globální strategie podniku (GST), informační strategie (IST), úvodní studie (UST), globální analýza a návrh (GAN), detailní analýza a návrh (DAN), implementace IS (IMP), zavádění (ZAV), provoz a údržba (PÚ). První dvě fáze se týkají IS jako celku. Ostatní fáze se týkají jednotlivých informatických projektů, které vytvářejí novou aplikaci (komponentu) IS, resp. novou verzi aplikace. Životní cyklus aplikace se uzavírá tehdy, kdy už není efektivní přizpůsobovat danou aplikaci novým podmínkám a novým požadavkům pomocí údržby. V tom případě je aplikace vyměněna za novou, jejíž vývoj zahajuje nový cyklus.