Tento text je určen pro studijní účely a je částí knihy

Metodiky budování informačních systémů

kategorizace, agilní metodiky, vzory pro návrh metodiky

autor

Alena Buchalcevová

citace

BUCHALCEVOVÁ, Alena. Metodiky vývoje a údržby informačních systémů. 1. vyd. Praha : Grada, 2005. 163 s. Management v informační společnosti. ISBN 80-247-1075-7.

Obsah

OBSAH 1

1 ÚVOD 3

1.1 Použitá terminologie 4

1.2 Typografické konvence 6

2 SOUČASNÝ STAV METODIK BUDOVÁNÍ IS/ICT 7

2.1 Faktory ovlivňující metodiky budování IS/ICT 7

2.2 Specifika budování IS/ICT 8

2.2.1 Složitost vývoje software 8

2.2.2 Procesy vývoje software jako empirické procesy 9

2.2.3 Vývoj software jako kooperativní hra 9

2.2.4 Software jako metaprodukt 10

2.3 Stav v oblasti metodik v ČR a ve světě 10

3 KATEGORIZACE METODIK 12

3.1 Objektivní příčiny existence různých metodik budování IS/ICT 12

3.2 Kritérium Zaměření metodiky 13

3.3 Kritérium Rozsah metodiky 14

3.4 Kritérium Váha metodiky 16

3.5 Kritérium Typ řešení 17

3.6 Kritérium Doména 17

3.7 Kritérium Přístup k řešení 19

3.8 Metapopis metodiky 20

4 RIGORÓZNÍ METODIKY 22

4.1 Model zralosti SW 22

4.1.1 Identifikace metodiky a zdrojů 22

4.1.2 Definice základních pojmů 23

4.1.3 Charakteristika metodiky 23

4.1.4 Úrovně zralosti 24

4.1.5 Personal Software Process a Team Software Process 26

4.1.6 Hodnocení metodiky 26

4.2 Metodika OPEN 27

4.2.1 Identifikace metodiky a zdrojů 27

4.2.2 Charakteristika metodiky 27

4.2.3 Hodnocení metodiky 28

4.3 Metodika Rational Unified Process 29

4.3.1 Identifikace metodiky a zdrojů 29

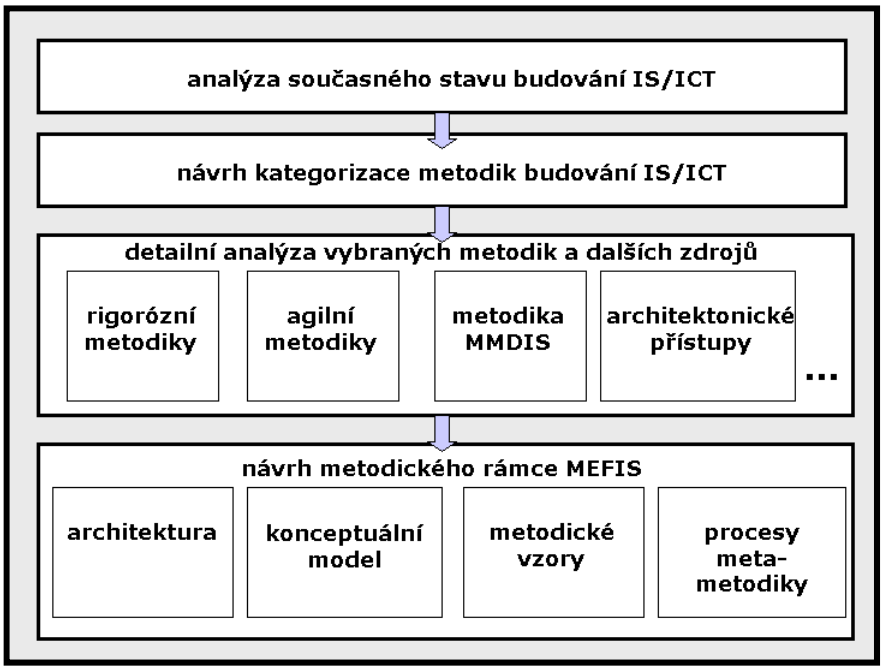
4.3.2 Charakteristika metodiky 29

4.3.3 Hodnocení metodiky 30

4.4 Metodika Enterprise Unified Process **31**
4.4.1 Identifikace metodiky a zdrojů 31
4.4.2 Charakteristika metodiky 31
4.4.3 Hodnocení metodiky 32
5 AGILNÍ METODIKY **33**
5.1 Hlavní principy agilních metodik **33**
5.2 Dynamic Systems Development Method (DSDM) **35**
5.2.1 Identifikace metodiky a zdrojů 35
5.2.2 Charakteristika metodiky 35
5.2.3 Hodnocení metodiky 36
5.3 Adaptive Software Development (ASD) **37**
5.3.1 Identifikace metodiky a zdrojů 37
5.3.2 Charakteristika metodiky 37
5.3.3 Hodnocení metodiky 38
5.4 Lean development **38**
5.4.1 Identifikace metodiky a zdrojů 38
5.4.2 Charakteristika metodiky 38
5.4.3 Hodnocení metodiky 40
5.5 Feature-Driven Development (FDD) **40**
5.5.1 Identifikace metodiky a zdrojů 40
5.5.2 Charakteristika metodiky 41
5.5.3 Hodnocení metodiky 41
5.6 Crystal metodiky **42**
5.6.1 Identifikace metodiky a zdrojů 42
5.6.2 Charakteristika metodiky 42
5.6.3 Hodnocení metodiky 42
5.7 Scrum **43**
5.7.1 Identifikace metodiky a zdrojů 43
5.7.2 Charakteristika metodiky 43
5.7.3 Hodnocení metodiky 44
5.8 Extrémní programování (XP) **45**
5.8.1 Identifikace metodiky a zdrojů 45
5.8.2 Charakteristika metodiky 45
5.8.3 Hodnocení metodiky 46
5.9 Agilní modelování **46**
5.9.1 Identifikace metodiky a zdrojů 46
5.9.2 Charakteristika metodiky 47
5.9.3 Hodnocení metodiky 49
5.10 Porovnání rigorózních a agilních metodik **50**
6 ARCHITEKTURA IS/ICT **53**
6.1 Charakteristika architektury IS/ICT **53**
6.2 Modelem řízená architektura **54**
6.3 Architektura orientovaná na služby **57**
SEZNAM POUŽITÉ LITERATURY **59**

1 Úvod

Jak už název napovídá, kniha se zabývá metodikami budování informačních systémů a informačních a komunikačních technologií (IS/ICT). Charakterizuje současný stav v oblasti metodik budování informačních systémů ve světě i v České republice, definuje kritéria kategorizace existujících metodik, analyzuje a kategorizuje nejvýznamnější současné metodiky a architektonické přístupy. Zároveň je zde představen návrh metodického rámce pro budování informačního systému firmy, který odráží současné trendy v informačních technologiích i metodických přístupech. Metodický rámec je chápán jako uspořádaná skupina metodik, respektive metodických vzorů, které jsou zaměřeny nejen na vývoj nového informačního systému, ale i na rozvoj stávajícího informačního systému a nasazování typového aplikačního softwarového vybavení. Metodické vzory pokrývají celý životní cyklus informačního systému od globální strategie podniku až po provoz a údržbu a zahrnují i procesy na úrovni celé organizace. Zároveň jsou specializovány na různé problémové domény (například systémy ERP, Business Intelligence, workflow, elektronické podnikání a další) a různé typy projektů. Strukturu knihy zachycuje obrázek 1.1.



obrázek 1.1: Struktura knihy

Současná situace v metodikách budování informačních systémů je analyzována v kapitole 2. Na základě zjištění, že v české ani světové odborné literatuře neexistuje odpovídající kategorizace metodik spojených s vytvářením informačních systémů, jsou navržena kritéria umožňující metodiky kategorizovat. Popis těchto kritérií a kategorie metodik jsou uvedeny v kapitole 3. Navržená kategorizace metodik a struktura metapopisu metodiky umožňují jednotně přistupovat k detailní analýze metodik. V současnosti se v odborné literatuře věnované metodickým přístupům k vývoji informačních systémů i na praktických projektech můžeme setkat se dvěma hlavními metodickými

proudy, které jsou označovány jako rigorózní metodiky a agilní metodiky. V kapitolách 4 a 5 jsou analyzovány a porovnány nejvýznamnější metodiky obou skupin. Kapitola 6 se zabývá současnými architektonickými přístupy. Kapitola **Chyba! Nenalezen zdroj odkazů.** obsahuje popis metodického rámce. Definuje základní charakteristiky metodického rámce, jeho architekturu, prvky a konceptuální model. Definuje klasifikaci metodických vzorů. Obsah metodického vzoru je dokumentován na příkladě vzoru M02 *Nový objektově orientovaný vývoj obecného software vlastními silam*, který je uveden v kapitole **Chyba! Nenalezen zdroj odkazů.**. V kapitole **Chyba! Nenalezen zdroj odkazů.** jsou definovány principy vytvoření metodiky a procesy při odvození metodiky pro konkrétní projekt.

1.1 Použitá terminologie

Definice použitých termínů a zkratek jsou uvedeny souhrnně na konci knihy v kapitole *Seznam použitých pojmů a zkratek*. Oblast IS/ICT nemá v některých případech obecně uznávané české ekvivalenty pro vybrané anglické termíny. V těchto případech je proto uveden anglický ekvivalent termínu v závorce za jeho českým překladem, některé názvy metodik jsou ponechány v angličtině. Protože terminologie v oblasti informačních systémů i metodik pro jejich vytváření není ustálená a jednotná, jsou v této v této podkapitole vymezeny základní termíny tak, aby při jejich použití v dalším textu byl jasný jejich význam. Jedná se jak o termíny v oblasti předmětu metodik, tedy budovaného systému, tak termíny týkající se metodik samotných.

Základním termínem v oblasti předmětu metodik je informační systém (respektive zkratka IS/ICT). „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)“ [KIT,2003]. V současnosti, kdy se IS/ICT stávají integrální součástí podnikových procesů, se můžeme setkat zejména v anglicky psané odborné literatuře s novými termíny „celopodnikové systémy na bázi ICT“ (Enterprise ICT Systems), „celopodniková softwarově intenzivní řešení“ (Enterprise Software Intensive Solutions) a dalšími. Tyto termíny jsou velmi výstižné v anglickém jazyce, ale obtížně se překládají do češtiny a jejich české překlady se zatím dostatečně nevžily. Proto je v knize používán zažitý termín informační systém respektive zkratka IS/ICT. Informační systém zahrnuje jak automatizované, tak neautomatizované činnosti. Automatizované činnosti podporuje software, tedy programové vybavení. V anglicky psané odborné literatuře je pojem software (či zkratka SW) používán často. V české odborné literatuře se někdy místo pojmu software používá pojem programový systém. Programovým systémem je chápán softwarový produkt, který je tvořen množinou programových jednotek (modulů, objektů, komponent, služeb) a jejich vzájemných vazeb [Buchalcevová,Stanovská,Šimůnek,2003]. Při popisu existujících zahraničních metodik jsou použity termíny, které se v těchto metodikách používají. Tyto termíny jsou vysvětleny při popisu jednotlivých metodik.

V oblasti metodik samotných je třeba vymezit především pojem metodika. Metodika (methodology) představuje v obecném smyslu souhrn metod a postupů pro realizaci určitého úkolu. Kniha je zaměřena na metodiky, které se zabývají vývojem a údržbou informačního systému. Tyto metodiky bývají označovány v některých zdrojích jako metodiky vývoje IS/ICT. Protože v dnešní době je kromě vývoje nového řešení důležité také nasazení hotového řešení, rozšíření řešení a integrace stávajících řešení, jsou tyto metodiky označovány jako **metodiky budování IS/ICT.** Definice pojmu metodika budování IS/ICT vychází z definice uvedené v metodice MMDIS, která je blíže charakterizována v podkapitole **Chyba! Nenalezen zdroj odkazů.**. Podle ní je „metodika tvořena obecně uznávanými postupy a návody, které popisují činnosti při analýze, návrhu, vývoji, nasazování software stejně jako činnosti spojené s řízením projektu. Cílem metodiky je formalizovat postupy, definovat zodpovědnosti a pravidla komunikace“ [Voříšek,1997]. Tato definice je rozvedena například v [Řepa,1999] a vymezuje metodiku tvorby informačního systému jako „doporučený souhrn etap, přístupů, zásad, postupů, pravidel, dokumentů, řízení, metod, technik a nástrojů pro tvůrce informačních systémů, který pokrývá celý životní cyklus informačních systémů. Metodika by se měla vztahovat na všechny prvky informačního systému, na pracovníky, data, software, hardware, organizační procedury, ekonomické otázky spojené s vývojem a provozem systému, doporučené dokumenty, způsoby řízení v jednotlivých fázích životního cyklu systému.“ Podle této definice se metodika tvorby informačního systému zaměřuje na oblast spojenou jak s vývojem, tak s provozem informačního systému. Vývoj a provoz informačního systému je obtížné oddělit, neboť některé části informačního systému (subsystémy, moduly, komponenty, služby aj.) jsou v daném okamžiku v provozu, některé jsou rozvíjeny, některé jsou vytvářeny nově a všechny musí být dohromady integrovány. Proto metodiky budování IS/ICT pokrývají jak oblast vývoje, tak oblast provozu IS/ICT. Problematikou provozu IS/ICT se však nezabývají v celé šíři, neboť je tato oblast řešena v rámci metodik pro řízení informatiky (například COBIT, ITIL). Termín metodika budování IS/ICT je vymezen následovně:

Metodika budování IS/ICT definuje principy, procesy, praktiky, role, techniky, nástroje a produkty používané při vývoji, údržbě a provozu informačního systému, a to jak z hlediska softwarově inženýrského, tak z hlediska řízení.

Kromě pojmu metodika se můžeme setkat s pojmy proces a softwarový proces. Mnohé metodiky mají slovo „proces“ přímo ve svém názvu. Příkladem jsou metodiky *Rational Unified Process, Open Process, Object-Oriented Software Process* a další (viz kapitola 4). Existují metodiky hodnocení softwarových procesů, mluví se o zlepšování softwarových procesů apod. Softwarový proces je v kontextu těchto přístupů definován jako sada činností, metod, praktik a transformací, které lidé používají pro vývoj a údržbu software a dalších s tím spojených produktů (projektových plánů, návrhů, testovacích případů apod.) [Paulk]. Pojem softwarový proces je v této knize používán jen v rámci popisu zahraničních metodik, které tento pojem zavádějí (například *Capability Maturity Model* viz. podkapitola 4.1).

Ještě vymezíme pojem metodický rámec. **Metodický rámec (Methodology framework) je kolekce metodických vzorů pro různé domény, typy řešení a způsoby řešení spolu s principy a procesy pro vytvoření konkrétní metodiky.**

1.2 Typografické konvence

V textu se často vyskytují odkazy na různé metodiky a jejich prvky (jako například fáze, dimenze, procesy a další). Protože se jedná o názvy, jsou uváděny s velkým písmenem a kurzívou, aby byly dobře odlišitelné od ostatního textu.

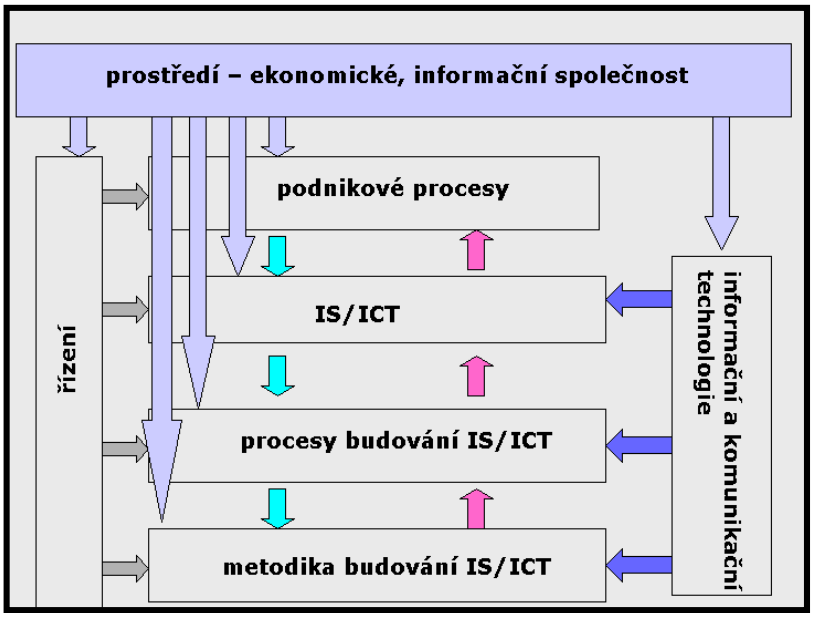
2 Současný stav metodik budování IS/ICT

Smyslem této kapitoly je charakterizovat současný stav v oblasti metodik budování IS/ICT a faktory, které jej ovlivňují. Mnohé rysy metodik a problémy s jejich aplikací jsou dány zvláštnostmi vývoje software, které jsou v této kapitole stručně charakterizovány.

2.1 Faktory ovlivňující metodiky budování IS/ICT

Oblast budování IS/ICT můžeme chápat jako součást podnikové informatiky, kterou můžeme vymezit jako „systém informačních a komunikačních technologií, dat a lidí, jehož cílem je efektivní podpora informačních a rozhodovacích procesů a procesů správy a využívání znalostí na všech úrovních řízení podniku“ [Vodáček,Rosický,1997]. Podniková informatika představuje souhrn zdrojů, procesů a služeb IS/ICT. Mezi zdroje můžeme zařadit zejména technologickou infrastrukturu (hardware, počítačová síť, základní software), aplikační software, data, spotřební materiál a obslužný personál a další. Procesy informatiky jsou procesy, které jsou spojeny s IS/ICT jako například „analýza a návrh“, „řízení konfigurací", „řízení projektů" apod. Služby informatiky představují definované rozhraní, které propojuje procesy informatiky s podnikovými procesy [Novotný,2003].

Chceme-li pochopit význam a úlohu metodik budování informačních systémů, je třeba se podívat na současný stav jak v oblasti IS/ICT, tak v celé společnosti. Turbulentní změny prostředí, globalizace ekonomiky, rostoucí konkurence představují vnější vlivy. Organizace mění ale i způsob řízení. Direktivní řízení je nahrazováno vůdcovstvím, prosazují se procesní přístupy, klíčovou roli začíná hrát řízení znalostí a řízení změn. Tyto faktory, které ovlivňují fungování podniků a dalších organizací, zprostředkovaně ovlivňují i IS/ICT. Dalším faktorem působícím na metodiky je velmi rychlý vývoj informačních a komunikačních technologií. Budování IS/ICT je ovlivněno i úrovní stávajícího informačního systému. Informační systémy dnes nevznikají na zelené louce, ale musí integrovat stávající systémy nejen v rámci podniku, ale i mezi podniky. Všechny tyto faktory zachycuje obrázek 2.1.



obrázek 2.1:Faktory ovlivňující metodiky budování IS/ICT

2.2 Specifika budování IS/ICT

Je třeba poznamenat, že většina metodik v oblasti IS/ICT i odborná literatura se zaměřuje na vývoj nového informačního systému. Přitom v dnešní době je hlavním úkolem zejména rozvoj stávajících systémů, implementace typových programových řešení, integrace dílčích řešení do celopodnikového systému. Touto oblastí se většinou metodiky nezabývají, i když se v poslední době začínají objevovat články a publikace na toto téma. Ať jde o nový vývoj, rozvoj, implementaci typového řešení, cílem a výsledkem je vždy implementované programové vybavení – software. Tato kapitola je zaměřena právě na software a jeho zvláštnosti jako produktu i zvláštnosti jeho vývoje (software development). Na celou historii vývoje software, která není ve srovnání s ostatními odvětvími dlouhá, můžeme pohlížet jako na boj se složitostí. Na jedné straně se do tohoto boje nasazují stále výkonnější nástroje, na druhé straně rostou požadavky na software (rozsah, kvalita, rychlost vývoje, flexibilita, přívětivost a další). Hlavním atributem software je tedy stále složitost jeho vývoje, která je také jednou z příčin velkého počtu neúspěšných softwarových projektů. V následujících odstavcích se zamyslíme nad některými zvláštnostmi vývoje software, které by měly metodiky budování IS/ICT zohlednit.

2.2.1 Složitost vývoje software

Na vývoj software má vliv jak prostředí vývoje, tak cílové prostředí. Proměnnými veličinami při vývoji software jsou dle [Scrum2]:

- dostupnost kvalifikovaných specialistů (pro nové technologie, nástroje, metody a domény je malý počet kvalifikovaných odborníků),
- stabilita technologie pro implementaci (nové technologie jsou méně stabilní),
- stabilita a schopnosti nástrojů,

- efektivnost používaných metod,
- dostupnost expertů na věcnou oblast i technologii,
- nová funkcionalita a její vztah k existující funkcionalitě,
- metodika a její flexibilita,
- konkurence,
- čas,
- zdroje,
- další proměnné.

Celková složitost vývoje software je funkcí těchto proměnných, přičemž tyto proměnné se v průběhu projektu mění:

složitost = f(proměnné prostředí vývoje + proměnné cílového prostředí)

Roste-li složitost projektu, je třeba zařazovat do procesu více kontrolních prvků (například řídit rizika apod.).

2.2.2 Procesy vývoje software jako empirické procesy

Software má mnoho aspektů, které jej odlišují od jiných produktů, a proto je i proces jeho vývoje odlišný. Tradiční přístupy předpokládají, že procesy při vývoji software je možné plně definovat a konzistentně opakovat. To předpokládá, že je možné definovat a opakovat:

- problém,
- řešení,
- nositele řešení – vývojáře,
- prostředí.

Tyto předpoklady však dle [Scrum2] a zastánců agilních přístupů při vývoji software neplatí. V mnoha případech není možné definovat problém na začátku projektu, protože požadavky nejsou přesně specifikovány a nebo se mění. Opakovatelnost řešení předpokládá, že je možné plně specifikovat architekturu a návrh. Také vývojáři nejsou stejní, ale liší se svými schopnostmi a znalostmi. Liší se i prostředí, ve kterém vývoj probíhá. Vývoj software tak probíhá v podmínkách chaosu a je to velmi složitý proces. Nelze jej tedy předem plně popsat, ale je nutné jej průběžně monitorovat a přizpůsobovat se změnám. Vývoj software tedy podle zastánců agilních přístupů není definovaný proces, ale **proces empirický**.

2.2.3 Vývoj software jako kooperativní hra

Tradiční rigorózní přístupy pohlížejí na vývoj software jako na inženýrskou disciplínu. Alistair Cockburn [Cockburn,CGM] pohlíží na vývoj software jako na kooperativní hru s omezenými zdroji

založenou na invenci a komunikaci. Jako příklad kooperativní hry uvádí tým horolezců. Jejich hra má jasný cíl, je kooperativní a konečná. Podobně je to i s vývojem software. Primárním cílem při vývoji software je dodávka software, který splňuje požadavky uživatelů. Sekundárním cílem je připravit se pro další hru, kterou je rozvoj systému a nebo vývoj nového systému. Pokud neuspějeme v primárním cíli, je sekundární cíl bezpředmětný. Z toho pohledu nemá smysl dodávat perfektní dokumentaci, když nedodáme fungující software. Na druhé straně ale úspěšné splnění primárního cíle nemusí znamenat úspěch v sekundárním cíli, tedy úspěch v dalších hrách.

2.2.4 Software jako metaprodukt

Velmi zajímavou charakteristiku software uvedl Clemens Szyperski, významný odborník v oblasti komponentového vývoje, ve svém článku pro LogOn Experts´Corner [Szyperski,11/2002]: „Dodávka software je spíše než dodávkou finálního produktu dodávkou plánu pro produkt. Na počítače se můžeme dívat jako na plně automatizované továrny, které přijímají tyto plány a vytvářejí z nich instance. V tomto smyslu je software **generický metaprodukt,** který může být použit pro vytvoření celé rodiny instancí**.“**

2.3 Stav v oblasti metodik v ČR a ve světě

Úspěšnost softwarových projektů není uspokojivá. Dle výsledků výzkumu společnosti Standish Group splňovalo v roce 2000 kritéria úspěšnosti jen 28% všech projektů na vývoj aplikací. [Johnson,2001]. Úspěšnost byla přitom definována tak, že projekt je dokončen včas, dle rozpočtu a se všemi specifikovanými funkcemi. Mezi 10 nejdůležitějšími faktory, které ovlivňují úspěšnost IS/ICT projektů, je i používání formální metodiky. Význam používání metodiky při budování IS/ICT dokumentuje také zpráva „2003 Worldwide IT Benchmark Report“ společnosti META Group [Metagroup,2002], která uvádí, že 51,6% všech respondentů používá při vývoji informačních systémů metodiku. Metodik, které se zabývají budováním IS/ICT je ale velké množství. Problém spočívá v tom, že nejsou dostatečně a jednotně popsány ani rozumně kategorizovány. Většinou se jedná o metodiky zaměřené jen na určitou fázi budování informačního systému (například objektově orientovaný návrh systému), na určitou věcnou oblast, na určitý typ projektu a podobně. Zároveň nejsou definována kritéria pro výběr vhodné metodiky pro určitý typ projektu ani postupy pro její přizpůsobení na konkrétní podmínky firmy a projektu. V poslední době se myšlenky na výběr vhodné metodiky a její přizpůsobení začínají ve světě objevovat (například [Cockburn,MetPerProj], [Ambler,DP], [Highsmith,2002]). V podmínkách České republiky však nejsou tyto přístupy systematicky popsány ani používány. Také procento firem, které používají při vývoji software metodiky je v České republice nižší než ve světě. Důvody této skutečnosti mohou být různé, ale jsem přesvědčena, že mezi hlavní důvody patří:

- nedostatek českých metodik[1], neboť většina metodik je v angličtině a nejsou lokalizovány do češtiny,
- metodiky se zpravidla šíří na komerční bázi a české firmy nechtějí či nemohou vydávat prostředky na nákup metodik,
- problémy s aplikací metodik zejména, pokud jsou v angličtině.

Proto si myslím, že je třeba zejména v podmínkách ČR v oblasti metodik vývoje IS/ICT ještě mnoho udělat. A tato kniha by v tom měla pomoci.

[1] z českých metodik můžeme uvést například metodiky *Objektově orientované metodiky a technologie* (OOMT) viz [Drbal,1997], *Multidimensional Management and Development of Information System* (MMDIS) [Vorisek,1997], *Business Object Relation Modeling* (BORM) viz [Polák,Merunka,Carda]

3 Kategorizace metodik

Hlavní problém, se kterým se při zkoumání metodik budování IS/ICT setkáme, spočívá v tom, že metodik je velké množství, nejsou dobře kategorizovány ani jednotným způsobem popsány. Tato skutečnost velmi znesnadňuje vyhledání vhodné metodiky pro určitý typ projektu. V této kapitole jsou nejprve identifikovány objektivní příčiny existence různých metodik a na jejich základě je navržena kategorizace metodik budování IS/ICT.

3.1 Objektivní příčiny existence různých metodik budování IS/ICT

Existence různých metodik je objektivně dána těmito skutečnostmi:

1. **Různé technologie vyžadují různé techniky a metody.**
   Objektově orientované metodiky vyhovují projektům, které využívají objektově orientované technologie, datově orientované metodiky vyhovují pro vývoj datově orientovaných aplikací apod.
2. **Organizace se liší firemní kulturou.**
   Mnohé implementace metodik selhávají, protože nepočítají s firemní kulturou. Firemní kultura může být s některými přístupy v rozporu, jiné může naopak podporovat. Při implementaci metodiky v organizaci je třeba analyzovat její firemní kulturu.
3. **Každý jedinec je jedinečný.**
   Primárním faktorem při vývoji informačního systému jsou lidé. Metodiky však tuto skutečnost dostatečně nezohledňují. Lidé nejsou zaměnitelné součástky, každý má jiné znalostní zázemí, jiným způsobem dosahuje cíle. Proto není možné vytvořit jedinou metodiku vyhovující všem, ale metodiku je třeba přizpůsobit konkrétním lidem, jejich znalostem a schopnostem.
4. **Každý tým je jedinečný.**
   Jedinečnost jedinců nutně vede k jedinečnosti týmů.
5. **Projekty se liší velikostí týmu.**
   Pro relativně malý počet lidí stačí relativně malá metodika.
6. **Projekty se liší svou důležitostí.**
   Vytváření systému pro řízení letového provozu vyžaduje jinou metodiku než mzdová agenda a také metriky úspěchu těchto dvou projektů jsou odlišné. Možných klasifikací systémů z hlediska důležitosti je více. Jedna z možných je uvedena v podkapitole **Chyba! Nenalezen zdroj odkazů.**.
7. **Projekty se liší podle postavení produktu na trhu.**
   Pro produkt vstupující na trh se zpravidla nepoužívá žádná metodika a nebo jen velmi „lehká“ metodika. Naproti tomu pro produkt zavedený na trhu je možné aplikovat rigorózní metodiku

(viz kapitola 4) s přesně popsanými procesy. Produkt, který představuje konkurenční výhodu pro organizaci, je třeba vyvinout rychle a uplatní se při něm zpravidla agilní metodiky (popis agilních metodik je uveden v kapitole 5).

8. **Projekt existuje v rámci určitého specifického vnějšího prostředí.**
   Některé projekty musí odpovídat pravidlům pro státní zakázky, některé projekty jsou vázány na dodavatele a jeho požadavky, mají přesně stanovený rozpočet, čas apod.

Výše uvedené skutečnosti mají za následek existenci různých metodik budování IS/ICT. Jednotlivé metodiky se liší například podle toho, jakou část životního cyklu postihují, jak přesně a jakým způsobem definují jednotlivé kroky při budování informačního systému a dalšími hledisky. V následujících podkapitolách jsou popsána kritéria, která lze využít pro kategorizaci metodik budování IS/ICT:

1. **Zaměření metodiky**
2. **Rozsah metodiky**
3. **Váha metodiky**
4. **Typ řešení**
5. **Doména**
6. **Přístup k řešení**

3.2 Kritérium Zaměření metodiky

Základním kritériem rozlišení metodik budování IS/ICT je skutečnost, zda je metodika zaměřena na budování IS/ICT celé organizace a nebo jen na jednotlivý projekt. V rámci tohoto kritéria jsou definovány dvě kategorie metodik – globální metodiky a projektové metodiky (tabulka 3.1).

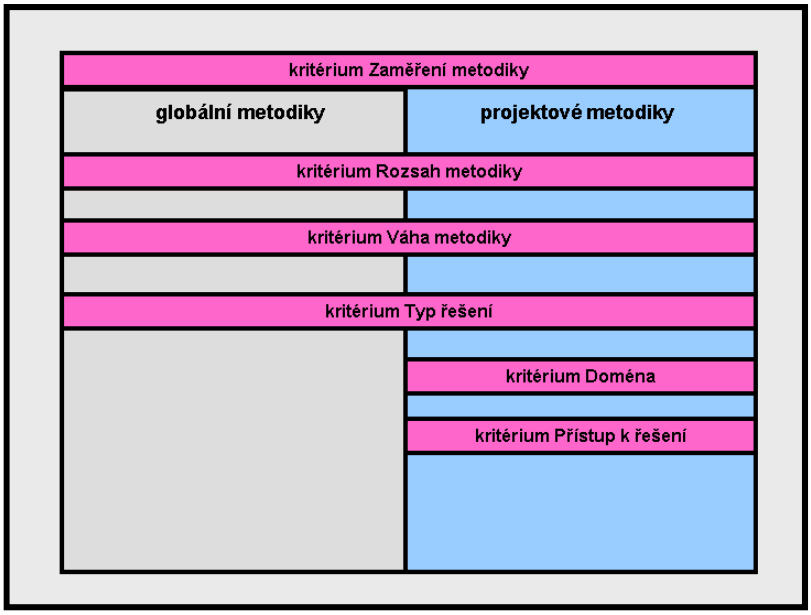
Kritérium Zaměření metodiky	
kód	*význam*
EM	**globální metodiky** (Enterprise Methodologies) metodiky zaměřené na vývoj, provoz i řízení celopodnikového IS/ICT
PM	**projektové metodiky** metodiky pro vývoj respektive zavedení informačního systému v určité oblasti

tabulka 3.1: Kritérium Zaměření metodiky

Většina publikovaných metodik patří do kategorie projektových metodik. Jsou to metodiky, které se zaměřují na vývoj, rozvoj či zavedení software v rámci jednoho projektu, pro určitý subsystém. Rostoucí význam integrace aplikací (Enterprise Application Integration, EAI) a globální architektury zvyšuje význam globálních metodik. Do kategorie globálních metodik můžeme zařadit metodiku MMDIS, která je od začátku 90 let rozvíjena na Katedře informačních technologií VŠE v Praze (viz. podkapitola **Chyba! Nenalezen zdroj odkazů.**). Mezi globální metodiky můžeme zařadit i *Model zralosti* (*Capability Maturity Model, CMM*) charakterizovaný v podkapitole 4.1. V oblasti

globálních metodik se významně angažuje organizace OMG (Object Management Group), která v roce 2001 přišla se svou strategickou iniciativou *Model Driven Architecture* (MDA), která je charakterizována v podkapitole 6.2.

Rozsah platnosti dalších kritérií vychází z rozdělení metodik na globální a projektové. Některá kritéria lze aplikovat jak na globální metodiky, tak na projektové metodiky (například kritérium *Rozsah metodiky* nebo *Váha metodiky*). Ostatní kritéria se aplikují v rámci kategorie projektových metodik respektive té části globální metodiky, která odpovídá projektovému řešení. Tuto situaci ilustruje obrázek 3.1.



obrázek 3.1: Působnost kritérií kategorizace metodik

3.3 Kritérium Rozsah metodiky

Rozsahem metodiky se zpravidla chápe počet fází životního cyklu informačního systému, které metodika pokrývá. Pro potřeby kategorizace metodik budování IS/ICT je rozsah metodiky budování IS/ICT definován jako průnik tří hledisek, kterými jsou:

- fáze životního cyklu,
- role,
- dimenze.

Toto vymezení rozsahu metodiky vychází z pojetí A. Cockburna [Cockburn,MetSpace], ale je upraveno o hledisko dimenzí. Rozsah fází životního cyklu určuje, kde metodika začíná a kde končí. Specifikace fází je převzata z metodiky MMDIS [Voříšek,1997] (viz tabulka 3.2). Budování IS/ICT je složitý proces, který vyžaduje spolupráci celé řady specialistů. Proto definují metodiky role jako typové skupiny pracovníků. Jednotlivé metodiky se liší počtem a typem rolí, které zahrnují.

Fáze životního cyklu	
zkratka	*název fáze*
GST	globální strategie
IST	informační strategie
UST	úvodní studie
GAN	globální analýza a návrh
DAN	detailní analýza a návrh
IMP	implementace
ZAV	zavádění
PUR	provoz a údržba

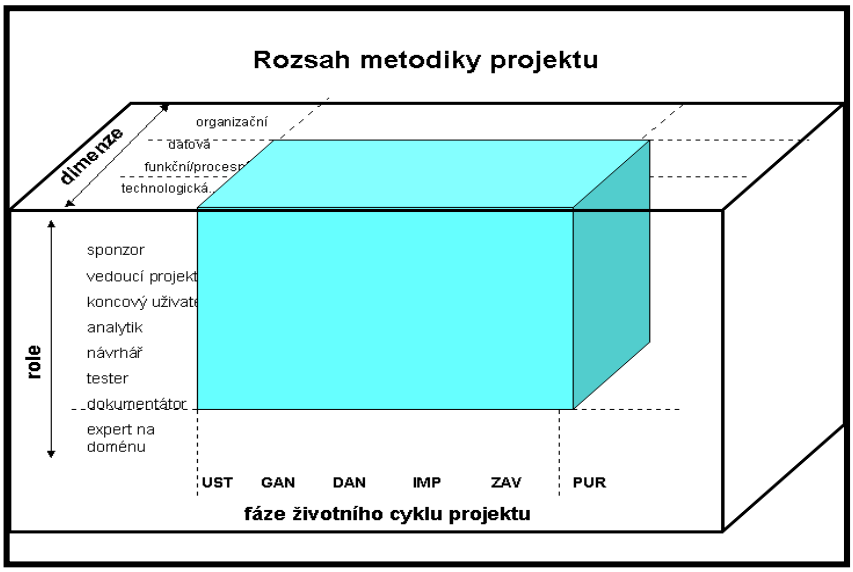
tabulka 3.2: Fáze životního cyklu

Některé metodiky se zabývají pouze jednou či několika málo dimenzemi. Vývoj software má však celou řadu aspektů a proto je účelné na něj pohlížet z různých dimenzí. S dimenzemi pracuje explicitně metodika MMDIS, která rozlišuje obsahové dimenze a dimenze, které se aplikují na metodiku samotnou – rozdělení do fází, dokumentační dimenze, metodická apod. Pro specifikaci kritéria *Rozsah metodiky* jsou uvažovány jen dimenze vyvíjeného systému a jsou oproti dimenzím definovaným v MMDIS upraveny. Přehled dimenzí použitých pro definici kritéria *Rozsah metodiky* uvádí tabulka 3.3.

Dimenze vyvíjeného systému		
dimenze	*zkratka*	*význam*
hardware	HW	typy, parametry a počty počítačů, periferních zařízení, komunikačních sítí a dalších technických prostředků
technologie	TECH	platforma (operační systém), architektura programového systému, SŘBD, technologie middleware apod.
data/informace	DATA	obsah a struktura datové základny a její fyzické uložení
funkce/procesy	FUN	procesy probíhající v podniku a možnost jejich podpory informačním systémem, funkce informačního systému
uživatelské rozhraní	UI	uživatelské rozhraní systému
pracovní	PRA	potřebná struktura pracovníků
organizační/legislativní	ORG	specifikace zákonů, norem a směrnic, které musí být při tvorbě IS/ICT respektovány
ekonomická	EKO	zahrnuje finanční náklady tvorby a provozu IS/ICT a přínosy podniku z užití IS/ICT, zdroje

tabulka 3.3: Specifikace dimenzí pro kritérium Rozsah metodiky

Kritérium *Rozsah metodiky* lze aplikovat jak na globální metodiky, tak na projektové metodiky. Schématické vyjádření náplně kriteria *Rozsah metodiky* pro projektovou metodiku zachycuje obrázek 3.2.



obrázek 3.2: Rozsah metodiky projektu

3.4 Kritérium Váha metodiky

V poslední době se můžeme setkat zejména v zahraniční odborné literatuře s pojmy „lehká metodika“ nebo „těžká metodika“. Tyto kategorie metodik jsou výsledkem aplikace kritéria *Váha metodiky*. Vymezení pojmu váha metodiky vychází z prací A.Cockburna, například [Cockburn,MetSpace], [Cockburn,MetPerProj], který zavádí charakteristiky metodiky, které označuje zkratkou **PARTS** – **p**recision, **a**ccuracy, **r**elevance, **t**olerance, **s**cale. Význam těchto charakteristik vysvětluje tabulka 3.4. Od těchto charakteristik jsou potom odvozeny pojmy velikost, hustota a váha metodiky. **Velikost metodiky** (methodology size*)* vyjadřuje počet kontrolních prvků obsažených v metodice. **Hustota metodiky (**methodology specific density*)* vyjadřuje míru podrobnosti a těsnost tolerance metodiky, požadovanou detailnost a konzistenci prvků. **Váha metodiky (**methodology weight) je potom součin velikosti metodiky a hustoty metodiky [Cockburn,MetSpace] .

Charakteristiky metodiky		
charakteristika	*angl. ekvivalent*	*význam*
podrobnost	*precision*	vyjadřuje, do jaké podrobnosti (hloubky) se metodika daným tématem zabývá
přesnost	*accuracy*	vyjadřuje, jak přesně je dané téma zpracováno
relevance	*relevance*	určuje, zda se metodika zabývá určitým tématem
tolerance	*tolerance*	určuje, jaké množství odchylek je povoleno
měřítko	*scale*	definuje míru zaostření (několik položek může být sbaleno do jediné)

tabulka 3.4: Charakteristiky metodiky

Rozvoj metodik v 80. a 90. letech 20. století vedl ke vzniku velmi propracovaných metodik s přesně definovanými procesy, činnostmi a artefakty. Tyto metodiky se označují jako **rigorózní** nebo „těžké metodiky“ a blíže jsou charakterizovány v kapitole 4. V poslední době se začínají prosazovat „lehké metodiky“, které definují místo podrobných procesů spíše principy a praktiky. Tyto metodiky jsou označovány jako **agilní** a jsou charakterizovány v kapitole 5.

3.5 Kritérium Typ řešení

Při realizaci informačního systému se nevyvíjí vždy nový software, ale často se implementuje typový aplikační software. Příkladem typového aplikačního software je například SAP R3, BAAN, SIEBEL a další. Je zřejmé, že metodika implementace typového aplikačního software se liší od metodiky vývoje software. V rámci metodiky pro vývoj software ovšem také existují odlišnosti. Vývoj software dnes neprobíhá zpravidla od začátku, ale často se rozšiřuje stávající řešení a nebo se provádí integrace řešení. V dnešní době se některé části IS/ICT zajišťují také formou Application Service Provision[2] či jiné formy outsourcingu[3], a proto je v rámci kritéria *Typ řešení* přidána hodnota „užití řešení". Navržené hodnoty kritéria *Typ řešení* uvádí tabulka 3.5.

Kritérium Typ řešení	
kód	*význam*
NEW	vývoj nového řešení (na zelené louce)
INT	integrace řešení
UPG	rozvoj a rozšíření řešení (upgrade)
TYP	customizace a implementace typového řešení
USE	užití řešení

tabulka 3.5: Kritérium Typ řešení

3.6 Kritérium Doména

Kritérium *Doména* se používá v rámci projektových metodik. Metodika pro vytvoření datového skladu je jiná než vybudování workflow či vývoj klasické aplikace. Doména představuje určitou předmětnou oblast, respektive určitou skupinu podnikových procesů, na jejichž podporu je informační systém vytvářen. Specifikace domén pro účely kategorizace metodik vychází z aplikační architektury IS/ICT, jejíž zobecněné schéma uvádí obrázek 3.3.

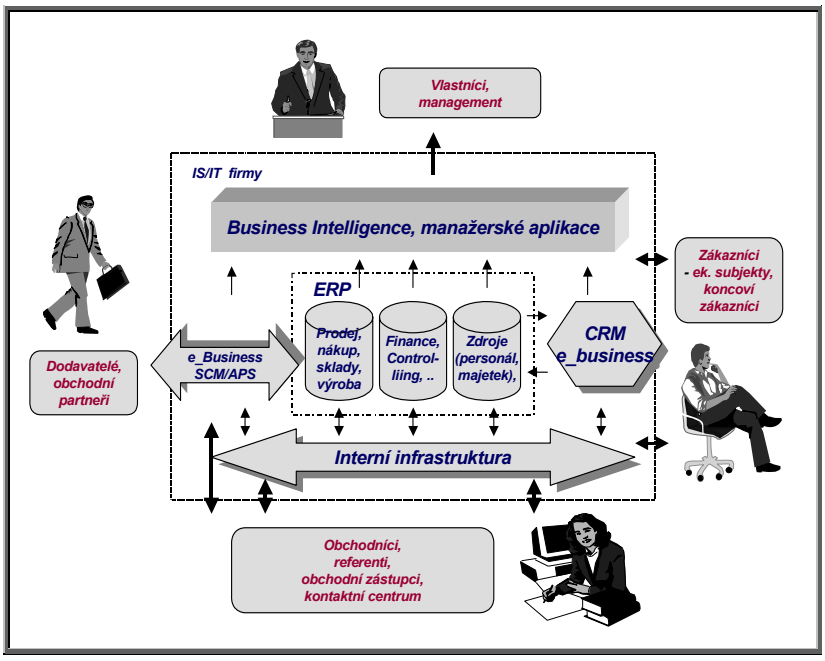
Obecné schéma architektury IS/ICT znázorňuje jednotlivé části informačního systému. V současné době se IS/ICT soustřeďuje kromě podpory procesů v organizaci zejména na řízení externích vztahů se zákazníky, dodavateli a dalšími partnery. Řízení vztahů se zákazníky, tzv. CRM (Customer Relationship Management), zahrnuje technickou a softwarovou podporu kontaktních center pro zákazníky, elektronickou podporu prodeje, podporu marketingu apod. Řízení dodavatelských řetězců SCM (Supply Chain Management) se zabývá řízením celého řetězce dodávky od dodavatele prvotních produktů a surovin přes celou řadu výrobních a obchodních mezičlánků až po dodávku konečnému zákazníkovi a má úzkou vazbu na aplikace progresivních plánovacích procedur – APS (Advanced Planning and Scheduling). Elektronické obchodování (e_commerce) představuje „podporu provádění obchodní transakce (nebo její části) prostředky IS/ICT (typicky i Internetem) a příslušnými

[2] Application Service Provision, poskytování aplikačních služeb. Jedna z forem outsourcingu. Specializovaná firma (Application Service Provider) na vlastní informační a komunikační technologii provozuje služby, které nabízí k použití externím zákazníkům. Zákazník je k aplikaci obvykle připojen přes Internet [KIT,2003].

[3] 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 [KIT,2003].

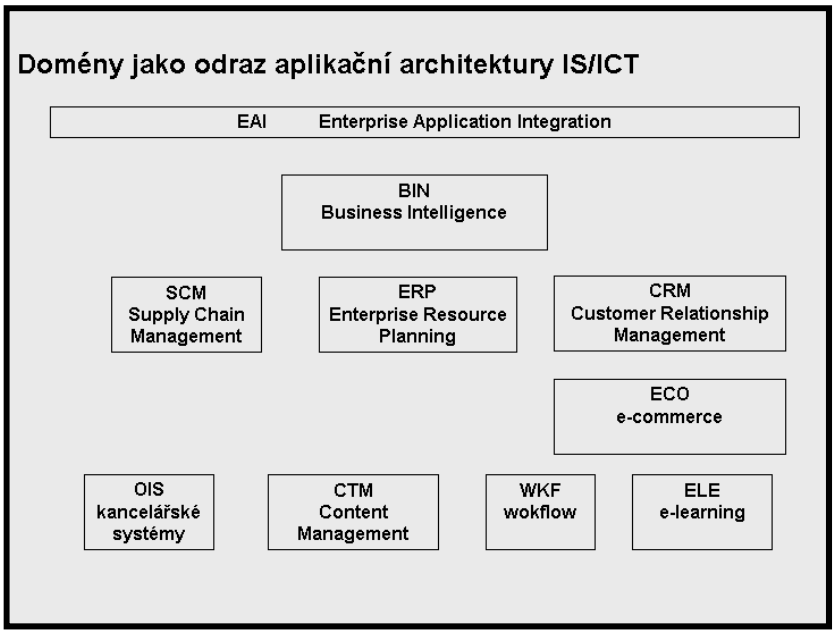
aplikačními programy (ASW)“ [KIT,2003]. Jádrem informačního systému jsou dnes již v jistém smyslu klasické integrované aplikační systémy – ERP (Enterprise Resource Planning), které pokrývají všechny nebo většinu oblastí podnikového řízení. Podporu všech základních řídících a administrativních operací podniku zajišťuje software pro interní infrastrukturu podniku – kancelářské systémy, aplikace pro řízení pracovních toků (workflow), aplikace a technologie pro správu dokumentů (Content management) a také elektronické vzdělávání (e–learning). Na nejvyšším místě je ve schématu umístěn blok tzv. business intelligence, který představuje komplex aplikací podporující analytické a rozhodovací aktivity vedoucích pracovníků podniku – manažerské aplikace, aplikace datových skladů a datových tržišť, dolování dat.

Na základě aplikační architektury IS/ICT (obrázek 3.3) a analýzy řady publikací, které se zabývají různými typy informačních systémů[4], jsou navrženy domény pro kategorizaci metodik (tabulka 3.6 a obrázek 3.4).



obrázek 3.3: Aplikační architektura IS/ICT firmy zdroj [Dohnal,Pour,1997]

[4] například [Basl,2002] , [Dohnal,2002], [Dohnal,Pour,1997]



obrázek 3.4: Domény IS/ICT

Kromě jednotlivých aplikačních domén je připojena ještě doména, EAI *Integrace podnikových aplikací*, která akcentuje integrační hledisko a zaměřuje se na vytvoření integrovaného celopodnikového systému. Protože jsou domény definovány pro potřeby kategorizace metodik, je připojena ještě doména obecný software, která zahrnuje blíže nespecifikovanou problémovou oblast.

Kritérium Doména		
kód	*název*	*význam*
BIN	**Business Intelligence**	datové sklady, analýzy dat, dolování dat
CRM	**Customer Relationship Management**	řízení vztahů se zákazníky
CSW	**obecný software**	tato doména zahrnuje software, u kterého nemá smysl zabývat se specifickými rysy
CTM	**Content Management**	řízení obsahu
EAI	**Enterprise Application Integration**	integrace podnikových aplikací
ECO	**e–commerce**	elektronické obchodování
ELE	**e–learning**	elektronické vzdělávání
ERP	**Enterprise Resource Planning**	ERP systémy
OIS	**Office Information System**	kancelářské systémy
SCM	**Supply Chain Management**	řízení dodavatelských řetězců
WKF	**wokflow**	automatizace podnikových procesů, oběh dokumentů

tabulka 3.6: Kritérium Doména

3.7 Kritérium Přístup k řešení

Kritérium *Přístup k řešení* má smysl aplikovat jen v rámci projektových metodik a to zejména pro nový vývoj. Toto kritérium zohledňuje základní paradigma, na kterém je metodika založena. V současnosti se můžeme setkat s těmito hlavními přístupy k vývoji software:

- strukturovaný vývoj,
- rychlý vývoj aplikací (Rapid Application Development, RAD),

- objektově orientovaný vývoj (zahrnuje i komponentový vývoj).

Určitý přístup může být aplikován buď ve všech hlavních vývojových fázích (analýza, návrh, implementace) a nebo je možné přístupy v jednotlivých fázích kombinovat. Běžně se například používá objektová analýza a RAD návrh a implementace. Na základě kombinace přístupů, které se nejčastěji v praxi používají, jsou definovány hodnoty kritéria *Přístup k řešení*, které uvádí tabulka 3.7.

Kritérium Přístup k řešení	
kód	*význam*
ST	**strukturovaný vývoj** vývoj založený na strukturované analýze, strukturovaném návrhu a strukturované implementaci
RS	**RAD vývoj se strukturovanou analýzou** analýza probíhá strukturovaně, návrh a implementace se provádí pomocí RAD nástrojů
RO	**RAD vývoj s objektovou analýzou** analýza se provádí objektově, ale návrh a implementace je pomocí RAD nástrojů
OO	**objektový vývoj ve všech fázích** provádí se objektově orientovaná analýza, objektově orientovaný návrh i objektová implementace s persistentními objekty
OR	**objektový vývoj s relační databází** provádí se objektově orientovaná analýza, objektově orientovaný návrh i objektová implementace, ale data jsou uložena v relační databázi

tabulka 3.7: Kritérium Přístup k řešení

3.8 Metapopis metodiky

Výše uvedená kritéria jsou základem metapopisu metodiky (tabulka 3.8). V této struktuře jsou popsány metodiky analyzované v kapitolách 4 a 5.

Struktura popisu metodiky		
položka	*význam, hodnoty číselníku*	
ID metodiky		
Název metodiky		
Zkratka		
Autoři		
Rok vzniku		
Zaměření		
	EM	globální metodika
	PM	projektová metodika
Rozsah		
	fáze	vyjmenovat fáze, které pokrývá GST, IST, UST, GAN, DAN, IMP, ZAV, PUR
	dimenze	vyjmenovat dimenze, které pokrývá HW, TECH, DAT, FUN, UI, PRA, ORG, EKO
	role	vyjmenovat role, které pokrývá
Váha metodiky		
	LM	lehká
	MM	střední
	HM	těžká
Typ řešení		

Struktura popisu metodiky		
	NEW	vývoj nového řešení (na zelené louce)
	INT	integrace řešení
	UPG	rozvoj a rozšíření řešení (upgrade)
	TYP	implementace typového řešení
	USE	užití řešení
Doména	určit doménu (např. CRM, ERP, BI,...)	
Přístup k řešení		
	ST	strukturovaný vývoj
	RS	RAD vývoj se strukturovanou analýzou
	RO	RAD vývoj s objektovou analýzou
	OO	objektový vývoj ve všech fázích
	OR	objektový vývoj s relační databází
Slovní charakteristika		
Poznámka		

tabulka 3.8: Struktura popisu metodiky

4 Rigorózní metodiky

V současnosti můžeme sledovat dva hlavní proudy v metodických přístupech, které jsou označovány jako rigorózní metodiky a agilní metodiky. Hlavním kritériem, které tyto dva proudy odlišuje, je kritérium *Váha metodiky*, ale liší se i dalšími hledisky. V této kapitole jsou charakterizovány rigorózní metodiky, které vycházejí z přesvědčení, že procesy při budování IS/ICT lze popsat, plánovat, řídit a měřit. Snaží se podrobně a přesně definovat procesy, činnosti a vytvářené produkty, a proto bývají často velmi objemné. Rigorózní metodiky jsou zpravidla založeny na sériovém (vodopádovém) vývoji. Existují ale také rigorózní metodiky založené na iterativním a inkrementálním vývoji[5] Příkladem těchto metodik jsou *OPEN*, *Rational Unified Process* (RUP), *Enterprise Unified Process* (EUP), které jsou v této kapitole charakterizovány.

V rámci rigorózních metodik tvoří samostatnou kategorii metodiky pro hodnocení softwarových procesů (*Software Process Assesment*). Jsou realizovány zejména v rámci projektu SPICE, který představuje hlavní mezinárodní iniciativu pro podporu vývoje mezinárodního standardu pro hodnocení softwarových procesů. Výsledky projektu byly publikovány jako standard ISO/IEC TR 15504:1998 – *Software Process Assesment* [SPICE]. Vznik tohoto standardu nebyl podnícen požadavky vývoje software, ale požadavky na akvizice softwarových firem. Pokračováním projektu SPICE je projekt OOSPICE (*Software Process Improvement and Cabability Determination for Object Oriented/Component Based Software Development*), který je financován z prostředků Evropské unie. Zaměřuje se na zlepšování softwarových procesů pro komponentový vývoj aplikací (*Component Based Development*, CBD). Metodiky hodnocení softwarových procesů jsou založeny na přesvědčení, že kvalita procesu určuje kvalitu produktu, a proto popisují postupy, které umožňují hodnotit úroveň zralosti procesů při vývoji software. Nejznámější z těchto metodik je *Model zralosti* (*Capability Maturity Model)*, který je charakterizován v následující kapitole.

4.1 Model zralosti SW

Nejznámějším příkladem hodnocení softwarových procesů je *Model zralosti SW* (*Capability Maturity Model for Software)* označovaný zkratkou SW–CMM.

4.1.1 Identifikace metodiky a zdrojů

Model zralosti SW byl vyvinut v Institutu pro softwarové inženýrství (Software Engineering Institute – SEI) za účelem hodnocení dodavatelů softwarových řešení pro ministerstvo obrany USA. V současné podobě a struktuře je dostupný od roku 1995. Kromě SW–CMM byly vytvořeny i další modely, zejména *Systems Engineering Capability Model* (SECM) a *Integrated Product Development*

[5] Iterativní vývoj představuje opakované (iterativní) provádění jednotlivých fází při vývoji IS. Výsledkem každé iterace je funkční verze systému. Současné metodiky doporučují velmi krátké iterace (dny). Iterativní vývoj může probíhat buď pro celý systém, jehož funkčnost se v jednotlivých iteracích rozšiřuje, a nebo ve spojení s inkrementálním vývojem (systém se vyvíjí po přírůstcích). [KIT,2003]

Capability Maturity Model (IPD–CMM). Organizace, které zaváděly různé modely hodnocení procesů, měly problémy s jejich integrací. Proto byl vytvořen *Integrační model zralosti (Capability Maturity Model Integration,* CMMI), který spojuje a rozšiřuje výše uvedené modely a podporuje dva způsoby zlepšování – průběžný a postupný. První verze CMMI modelu byla publikována v prosinci roku 2000, v současnosti je k dispozici verze 1.1 [CMMI]. V dalších podkapitolách je popisován model SW-CMM, který je již zaveden v celé řadě firem a je základem modelu SW–CMMI i metodik PSP a TSP , které jsou popisovány v podkapitole 4.1.5.

4.1.2 Definice základních pojmů

V tomto odstavci jsou definovány pojmy používané v rámci metodik hodnocení softwarových procesů. **Softwarový proces** je definován jako sada činností, metod, praktik a transformací, které lidé používají pro vývoj a údržbu software a dalších produktů (projektových plánů, návrhů, testovacích případů apod.). **Schopnost softwarového procesu** (Software Process Capability) popisuje okruh očekávaných výsledků, které mohou být dosaženy následováním softwarového procesu – je to nástroj pro určení výstupů budoucích projektů. **Výkon softwarového procesu** (Software Process Performance) představuje skutečné výsledky dosažené následováním softwarového procesu. Vztah mezi schopností a výkonem nemusí být přímo úměrný. Například přechod na novou technologii zvyšuje schopnost procesu, ale je spojen s nutností zaškolení pracovníků a může vést k poklesu výkonu procesu. **Úroveň zralosti** (Maturity Level) je dobře definované prostředí pro evoluční budování zralých softwarových procesů. **Zralost softwarového procesu** (Software Process Maturity) je taková úroveň procesů, při které je proces explicitně definován, řízen, měřen, kontrolován a je efektivní. Zralost softwarového procesu vede ke zvýšení produktivity a kvality a zvyšuje se výkon procesu. Když organizace dosáhne určité zralosti softwarového procesu, vede to k jeho institucionalizaci prostřednictvím politik, standardů a organizační struktury. **Hodnocení softwarového procesu** (Software Process Assesment) je hodnocení stavu softwarového procesu v organizaci. Provádí je vyškolený tým softwarových profesionálů s cílem určit procesy s nejvyšší prioritou a vytvořit organizační podporu pro jejich zlepšení. **Klíčová oblast procesu (**Key Process Area) je oblast, která má podstatný význam pro zlepšení softwarového procesu [Paulk].

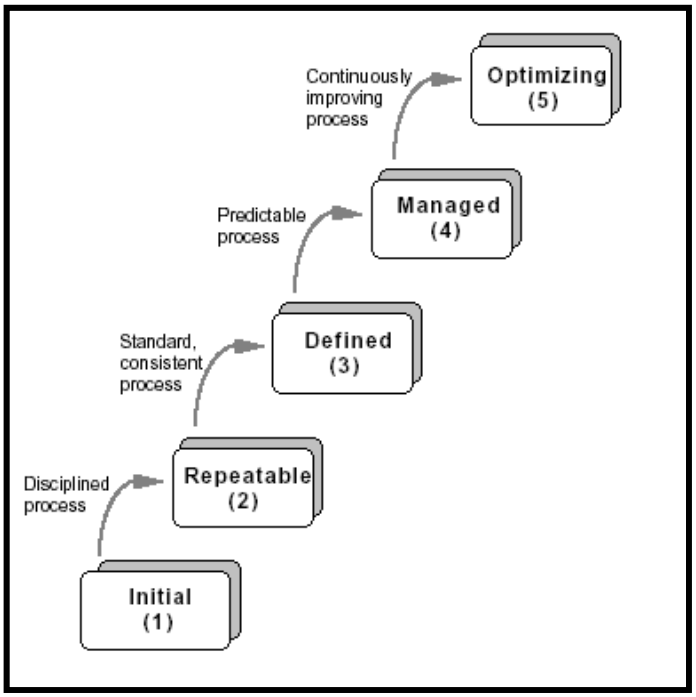
4.1.3 Charakteristika metodiky

Model SW–CMM definuje 5 úrovní zralosti softwarových procesů organizace. Na základě definované zralosti procesů se identifikuje několik nejdůležitějších oblastí pro zlepšení procesů. Přínos zavedení CMM je vysvětlován srovnáním nezralých a zralých softwarových procesů. Nezralé softwarové procesy jsou dle autorů CMM výsledkem improvizace. I když nějaké procesy existují, nelze je opakovat a předvídat, organizace pouze reaguje na vzniklé problémy a napravuje chyby. Plány a rozpočty jsou překračovány, protože nejsou založeny na realistických odhadech. Při snaze dodržet čas dochází ke snižování kvality. Činnosti, které mají kvalitu zvýšit se často z důvodu nedostatku času

neprovádějí. Na druhé straně zralá organizace je dle autorů CMM schopna řídit procesy při vývoji software a to v rámci celé organizace.

4.1.4 Úrovně zralosti

Neustálé zlepšování procesů je založeno na mnoha malých evolučních krocích. Stupňovitá struktura CMM představuje rámec, který organizuje evoluční kroky při zlepšování procesů do 5 úrovní zralosti tak, jak ukazuje obrázek 4.1. V následujících odstavcích je podrobněji charakterizováno chování organizací na jednotlivých úrovních zralosti.



obrázek 4.1: 5 úrovní zralosti SW–CMM zdroj [Paulk]

1. Počáteční úroveň (initial)

Softwarové procesy jsou náhodné a chaotické. Organizace na této úrovni zralosti nemají stabilní prostředí pro vývoj a údržbu software, reagují pouze na vzniklé problémy. Schopnost softwarových procesů na úrovni 1 nelze předpovídat, protože se procesy neustále mění a výsledky jsou ovlivněny schopnostmi a kvalifikací jednotlivců.

2. Opakovatelná úroveň (repeatable)

Organizace na této úrovni zralosti mají definovány a zavedeny postupy řízení projektu. Cílem úrovně 2 je zavedení efektivního řízení softwarových procesů. Efektivní softwarový proces je dokumentován, vyžaduje se jeho plnění, lidé jsou pro něj vyškoleni, je měřitelný a je možné jej zlepšovat. Manažeři sledují náklady, plán a funkcionalitu, problémy jsou identifikovány ihned při vzniku, jsou definovány projektové standardy a dohlíží se na jejich dodržování. Schopnost softwarových procesů na úrovni 2 může být definována jako disciplinovaná.

3. Definovaná úroveň (defined)

Organizace na této úrovni zralosti má definovány, dokumentovány a standardizovány procesy pro řízení i softwarově inženýrské činnosti, které jsou navzájem integrovány v rámci celé organizace. Je definován **Standardní softwarový proces organizace,** který je potom přizpůsobován na podmínky jednotlivých projektů**.** Tím vzniká **Softwarový proces projektu.** Zaměstnanci i manažeři jsou vyškoleni pro plnění rolí v procesech. Schopnost softwarových procesů na úrovni 3 může být definována jako standardní a konzistentní, protože jak činnosti inženýrské, tak činnosti řízení jsou stabilní a opakované.

4. Řízená úroveň (managed)

Organizace na této úrovni zralosti má stanoveny detailní metriky softwarových procesů i kvality produktu. Data z projektů se uchovávají v databázi a analyzují. Kontrola projektů je zajištěna odhalováním odchylek od definovaných mezí. Schopnost softwarových procesů na úrovni 4 může být definována jako predikovatelná, protože proces je měřitelný a operuje v rámci měřitelných mezí.

5. Optimalizovaná úroveň (optimizing)

Organizace na této úrovni zralosti má vytvořeny podmínky pro kontinuální zlepšování procesů. Organizace identifikuje slabé a silné stránky procesů předem ve snaze zabránit vzniku problémů. Data o efektivnosti softwarových procesů jsou využívána pro analýzy přínosů nových technologií a navrhovaných změn. Schopnost softwarových procesů na úrovni 5 může být definována jako kontinuálně se zlepšující.

Kromě úrovně 1 definuje každá úroveň zralosti několik klíčových oblastí procesů (Key Process Area, KPA), které určují procesy, na jejichž zlepšení by se měla organizace zaměřit. Klíčové oblasti procesů modelu SW–CMM uvádí tabulka 4.1. Každá KPA je popsána klíčovými praktikami, které zajišťují splnění cílů dané oblasti procesů. Klíčové praktiky popisují infrastrukturu a činnosti, které přispívají k nejefektivnější implementaci dané KPA.

úroveň	***klíčové oblasti procesů***
2	SW procesy spojené s vytvořením základních prvků **řízení projektu**: • řízení požadavků • plánování softwarových projektů • sledování softwarových projektů • řízení subdodávek • zajištění kvality software • řízení konfigurací
3	projekční i organizační hlediska procesů přes všechny projekty, tedy v **celé organizaci**: • zaměření na procesy na úrovni organizace • definice procesů na úrovni organizace • program školení • integrace software • softwarové inženýrství • koordinace mezi skupinami

úroveň	*klíčové oblasti procesů*
	• kontroly
4	vytvoření a **kvantitativní podpora** SW procesů a vytvářených SW produktů: • řízení kvantitativních procesů • řízení kvality
5	neustálé měřitelné **zlepšování SW procesů**: • prevence defektů • řízení technologických změn • řízení změn procesů

tabulka 4.1: Klíčové oblasti procesů modelu SW–CMM

Dosažení určité úrovně zralosti představuje zlepšení možností sledování stavu softwarových procesů. Pro organizaci na úrovni zralosti 1 představují softwarové procesy černou skříňku. Protože sled činností není dostatečně definován, manažeři mohou jen obtížně určit stav projektu. Na úrovni 2 je již zavedeno řízení projektu, což umožňuje zjišťovat stav v bodech přechodu - projektových milnících. Detailní sledování průběhu procesů není však stále možné. Na úrovni zralosti 3 je definován standardní softwarový proces organizace i způsob jeho přizpůsobení na podmínky projektu. Manažeři i inženýři znají své role a odpovědnosti v rámci procesu. Vedení se proaktivně připravuje na rizika, která mohou nastat. Na úrovni 4 jsou softwarové procesy kontrolovány kvantitativně. Manažeři mohou měřit pokrok a problémy a mají objektivní, kvantitativní základ pro rozhodování. Jejich schopnost předpovídat výsledky roste a proměnlivost procesů se zmenšuje. Na úrovni 5 dochází k řízenému zlepšování softwarových procesů.

4.1.5 Personal Software Process a Team Software Process

Při zavádění CMM měly organizace problémy s aplikací principů CMM. Proto Watts Humprey definoval v r.1989 metodiku *Personal Software Process* (PSP), která popisuje softwarově inženýrské procesy a praktiky, které mají zlepšit práci jednotlivců a podpořit procesy CMM [Humprey,PSP]. Na vyšších úrovních zralosti CMM se však ukázala potřeba zlepšit také práci týmů. Proto byla vytvořena metodika *Team Software Process* (TSP). TSP je plně definovaný a měřitelný proces, který mohou týmy používat pro plánování své práce, plnění plánů a kontinuální zlepšování procesů vývoje software. Nejdůležitější přínos TSP spočívá v tom, že pomáhá týmu řídit pracovní prostředí. Tým přebírá zodpovědnost za plánování své práce a vytvoření kvalitního produktu co nejrychleji a nejefektivněji [McAndrews].

4.1.6 Hodnocení metodiky

***Metapopis metodiky* Capability Maturity Model for Software**	
položka	*význam, hodnoty číselníku*
ID metodiky	MET1
Název metodiky	Capability Maturity Model for Software
Zkratka	SW–CMM
Autoři	Software Engineering Institute (SEI)

Metapopis metodiky **Capability Maturity Model for Software**		
Rok vzniku	1995	
Zaměření	**EM**	globální metodika
Rozsah	**fáze**	IST, UST, GAN, DAN, IMP, ZAV
	dimenze	TECH, DAT, FUN, PRA, ORG, EKO
	role	informační manažer, vedoucí projektu, analytik, vývojář
Váha metodiky	**HM**	těžká
Typ řešení	**NEW**	vývoj nového řešení (na zelené louce)
Doména	**CSW**	obecný SW
Přístup k řešení	nerozlišuje	
Slovní charakteristika	CMM – množina veřejných kritérií, která popisují charakteristiky zralé softwarové organizace a mohou být využita pro zlepšení procesů vývoje a údržby software nebo pro hodnocení rizika najímání subdodavatelů pro softwarový projekt.	
Poznámka		

CMM definuje množinu veřejných kritérií, která popisují charakteristiky zralé softwarové organizace. Tato kritéria mohou být využita pro zlepšení procesů vývoje a údržby software nebo pro hodnocení rizika najímání subdodavatelů pro softwarový projekt. CMM je metodika globální, neboť je zaměřena na procesy v rámci celé organizace, metodika „těžká“, neboť definuje podrobně softwarové procesy tak, aby mohly být opakovány. CMM se zaměřuje především na řízení softwarových procesů, v menší míře pak na softwarově inženýrské postupy. Definuje jednotlivé oblasti procesů a cíle, které mají být dosaženy. CMM tedy určuje „co“ by měly organizace dělat, ale nepopisuje způsob, „jak“ by to měly dělat. Způsob, jak naplnit cíle klíčových oblastí procesů definovaných v SW–CMM, je předmětem metodiky PSP na úrovni jednotlivců a TSP na úrovni týmů.

4.2 Metodika OPEN

4.2.1 Identifikace metodiky a zdrojů

Object–oriented Process, Environment and Notation je veřejně přístupná metodika podporující celý životní cyklus vývoje IS/ICT. Je zaměřena zejména na vývoj objektově orientovaných a komponentových aplikací. Metodika OPEN vyšla z projektu COMMA (*Common Object Methodology Metamodel Architecture*), jehož cílem bylo vytvořit metamodely různých metodik a metod objektově orientované analýzy a návrhu a navrhnout univerzální metodiku. Na vytvoření a rozšíření metodiky OPEN mají zásluhu zejména Brian Henderson–Sellers, Meilir Page Jones, Ian Graham a Donald Firesmith a další osobnosti organizované v konsorciu OPEN, mezinárodním seskupení více než 35 metodiků, vědeckých pracovníků, dodavatelů CASE nástrojů a vývojářů [OPEN].

4.2.2 Charakteristika metodiky

Metodika OPEN definuje procesní rámec, známý pod názvem *OPEN Process Framework* (OPF). Jde o procesní metamodel, ze kterého mohou být generovány instance specifické pro organizaci. Každá

taková instance je vytvořena výběrem činností, úloh a technik (tří hlavních tříd na metaúrovni) a jejich specifickou konfigurací. Tento proces je označován jako konstrukce procesu (process construction). Metodika dále rozlišuje přizpůsobení procesu (process tailoring), které spočívá **v** přizpůsobení činností a úloh tak, aby maximálně vyhovovaly problémové doméně. Tímto způsobem je OPEN flexibilní, může se přizpůsobit jak doméně, tak konkrétnímu projektu a zohlednit dovednosti členů týmu, kulturu organizace, požadavky specifické pro každou doménu. OPEN může být použit pro malé projekty, stejně jako pro velké, klíčové projekty. OPEN poskytuje podporu pro celý životní cyklus softwarové aplikace. Jeho součástí je řízení projektu a rámec pro znovupoužitelnost, podporuje modelování podnikových procesů, zaměřuje se na kvalitu software a použití metrik. Proto má těsné vazby s CMM a OOSPICE. OPF obsahuje velké množství metatříd, které jsou seskupeny do 5 hlavních skupin – pracovní jednotky, pracovní produkty, producenti, etapy a jazyky. Tyto metatřídy vytvářejí knihovnu komponent OPEN, ze které se vytvářejí a skládají instance metodiky. Pracovní jednotky jsou představovány činnostmi, úlohami a technikami. Činnosti (activities) určují, co je třeba udělat, ale neurčují jak. Činnosti se člení na úlohy (tasks), které mají takový rozsah, aby je mohl realizovat jeden vývojář nebo malý tým v relativně krátkém čase. Techniky potom určují, jak se úlohy provádějí. K těmto třem základním skupinám metatříd se připojují etapy různého typu (fáze, životní cykly, milníky) a jazyky (přirozený jazyk, modelovací jazyky, programovací jazyky). Pro modelování je možné zvolit UML (Unified Modeling Language) nebo OML (OPEN Modeling Language).

4.2.3 Hodnocení metodiky

***Metapopis metodiky* OPEN**		
položka	*význam, hodnoty číselníku*	
ID metodiky	MET2	
Název metodiky	Object–oriented Process, Environment and Notation	
Zkratka	OPEN	
Autoři	Brian Henderson–Sellers, Meilir Page Jones, Ian Graham a Donald Firesmith a další osobnosti organizované v konsorciu OPEN	
Rok vzniku	1997	
Zaměření	**PM**	projektová metodika
Rozsah	**fáze**	UST, GAN, DAN, IMP, ZAV
	dimenze	TECH, DAT, FUN, UI
	role	vedoucí projektu, uživatel, specialista na požadavky hlavní programátor,analytik, vývojář, tester
Váha metodiky	**HM**	těžká
Typ řešení	**NEW**	vývoj nového řešení (na zelené louce)
Doména	**CSW**	obecný SW
Přístup k řešení	**OO**	objektový vývoj ve všech fázích
	OR	objektový vývoj s relační databází
Slovní charakteristika	Metodika OPEN definuje procesní rámec – OPEN Process Framework (OPF). Jde o procesní metamodel, ze kterého mohou být generovány instance specifické pro organizaci. Metodika podporuje celý životní cyklus vývoje software, je zaměřena na procesy a je navržena pro vývoj objektově orientovaných a komponentových aplikací.	

Metapopis metodiky **OPEN**	
Poznámka	

Metodika OPEN je velmi podrobná metodika, která je ovšem zaměřena pouze na úroveň projektu a na objektově orientovaný a komponentový vývoj nového řešení. Přínosné jsou myšlenky vytváření konkrétní metodiky z metatříd, které jsou součástí OPF.

4.3 Metodika Rational Unified Process

4.3.1 Identifikace metodiky a zdrojů

Metodika *Rational Unified Process* (RUP) vznikla spojením přístupu Rational a metodiky *Objectory Process* Ivara Jacobsona. Původně byla nazvána *Rational Objectory Process*, ale v roce 1998 byla přejmenována na *Rational Unified Process*. *Rational Unified Process* je specializací obecnějšího procesu, který popsali Ivar Jacobson, Grady Booch a James Rumbaugh v knize „The Unified Software Development Process“ [Jacobson,Booch,Rumbaugh].

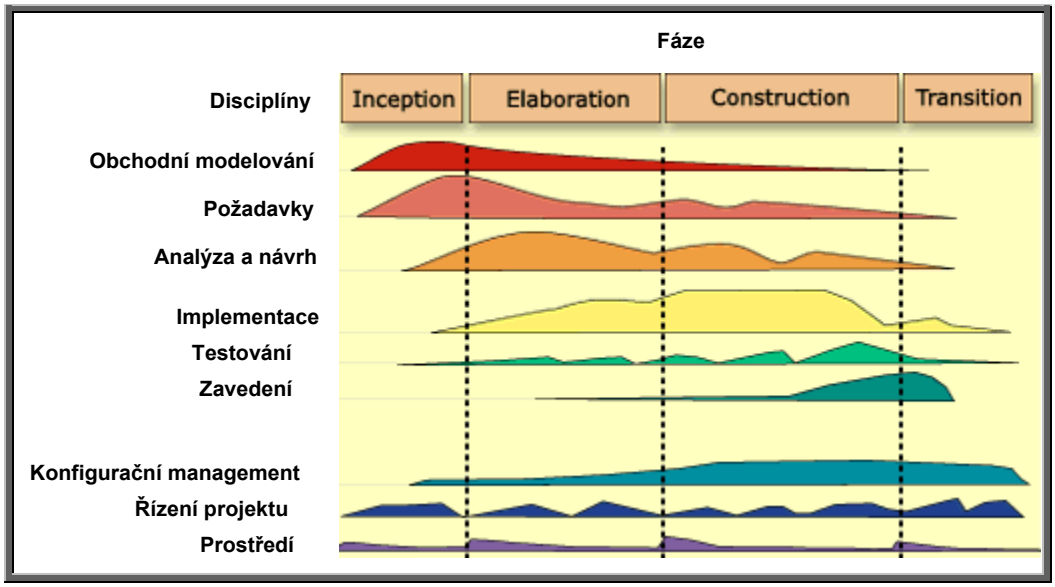
4.3.2 Charakteristika metodiky

Metodika *Rational Unified Process* je založena na tzv. nejlepších praktikách softwarového vývoje [RUP]:

- iterativní vývoj,
- řízení požadavků
- použití komponentové architektury,
- vizuální modelování,
- kontrola kvality software,
- řízení změn.

Proces vývoje software je možné popsat v rámci dvou dimenzí (osy na obrázku 4.2). Horizontální osa představuje dynamický pohled na proces, který je vyjádřen pomocí cyklů, fází, iterací a milníků. Vertikální osa reprezentuje statické hledisko procesu, popis činností, artefaktů, pracovníků a pracovních toků. Na vertikální ose znázorněno 9 tzv. disciplín[6], které představují logické seskupení činností definovaných v RUP. Graf ukazuje podíl jednotlivých disciplín v různých fázích projektu.

[6] později jsou v RUP označovány jako „core process workflows“



obrázek 4.2: Fáze a disciplíny RUP [RUP]

Životní cyklus software je rozdělen na cykly. Předmětem každého cyklu je nová verze produktu. Jeden vývojový cyklus je v RUP rozdělen do 4 po sobě jdoucích fází:

- *Počáteční fáze* (Inception),
- *Elaborační fáze* (Elaboration),
- *Konstrukční fáze* (Construction),
- fáze *Nasazení* (Transition).

Cílem *Počáteční fáze* je definice cílů projektu, požadavků, sestavení harmonogramu projektu (plán iterací), odhad nákladů projektu a definice rizik. Součástí této fáze může být vytvoření modelu nebo jednoduchého prototypu, na kterém se ověří, zda je možné se zvolenou technologií a pomocí zvolených nástrojů klíčové požadavky splnit. *Počáteční fáze* končí rozhodnutím, zda je projekt za daných požadavků, dostupných technologií, zdrojů a rozpočtu možné realizovat. Cílem *Elaborační fáze* je definovat architekturu systému. V této fázi by měl být vytvořen prototyp, který ověří všechny architektonické principy a umožní zpřesnění plánu realizace systému. Měly by být definovány komponenty, které je třeba vyvinout pro opakované použití. Obsahem *Konstrukční fáze* je návrh a realizace systému včetně testování. Prosazuje se pokud možno paralelní vývoj. Cílem fáze *Nasazení* je zajistit, aby uživatelé mohli systém používat. Součástí této fáze je školení uživatelů, předání dokumentace, vytvoření help–desk atd. Každá fáze je uzavřena milníkem – časovým okamžikem, ve kterém musí být splněny cíle fáze a dochází k rozhodování.

4.3.3 Hodnocení metodiky

***Metapopis metodiky* Rational Unified Process**	
položka	*význam, hodnoty číselníku*
ID metodiky	MET3
Název metodiky	Rational Unified Process

Metapopis metodiky **Rational Unified Process**		
Zkratka	RUP	
Autoři	Ivar Jacobson, Grady Booch, James Rumbaugh	
Rok vzniku	1998	
Zaměření	**PM**	projektová metodika
Rozsah	**fáze**	UST, GAN, DAN, IMP, ZAV
	dimenze	HW, TECH, DAT, FUN, UI
	role	vedoucí projektu, uživatel, specialista na požadavky, správce databáze, hlavní programátor, analytik, vývojář, dokumentátor tester, analytik testů, manažer testů, návrhář testů
Váha metodiky	**HM**	těžká
Typ řešení	**NEW**	vývoj nového řešení (na zelené louce)
Doména	**CSW**	obecný SW
Přístup k řešení	**OO**	objektový vývoj ve všech fázích
	OR	objektový vývoj s relační databází
Slovní charakteristika	Metodika Rational Unified Process je založena na tzv. nejlepších praktikách softwarového vývoje– iterativní vývoj, řízení požadavků, použití komponentové architektury, vizuální modelování, kontrola kvality software, řízení změn.	
Poznámka		

Metodika *Rational Unified Process* patří mezi rigorózní metodiky, neboť podrobně definuje procesy a činnosti při vývoji software. Zaměřuje se pouze na vývoj nového řešení realizovaný objektově orientovaným způsobem. Podstatným nedostatkem metodiky RUP je její zaměření pouze na úroveň projektu. Metodika má poměrně malý rozsah, neboť se zaměřuje pouze na vývoj řešení (nezahrnuje provoz a údržbu) a pouze na softwarově inženýrské role a dimenze. Silnou stránkou metodiky *Rational Unified Process* je její integrace s CASE nástroji firmy Rational, které podporují především oblast analýzy a návrhu, testování, řízení konfigurací a správu požadavků. Metodika RUP není podobně jako jiné zahraniční metodiky lokalizována do češtiny. K jejím výhodám patří, že je dodávána jako produkt, který zahrnuje znalostní bázi s webovým rozhraním a sadu nástrojů na přizpůsobení procesu a šablon do nástrojů firmy Rational. V současné době se tvůrci metodiky RUP snaží promítnout do ní některé myšlenky agilních metodik.

4.4 Metodika Enterprise Unified Process

4.4.1 Identifikace metodiky a zdrojů

Metodika *Enterprise Unified Process* (EUP) pochází z dílny Scotta W. Amblera a představuje rozšíření metodiky *Rational Unified Process* (RUP) [Ambler,AMT].

4.4.2 Charakteristika metodiky

Metodika *Enterprise Unified Process* rozšiřuje metodiku RUP ve dvou směrech. První směr představuje rozšíření na úroveň celé organizace. EUP definuje novou disciplínu – *Infrastructure Management*, která zahrnuje procesy realizované přes projekty. Druhý směr rozšíření metodiky RUP

představuje připojení fáze *Production,* jejíž náplní je provoz a údržba systému, a fáze *Retirement*, která popisuje činnosti nutné při odstranění produktu z používání [Ambler,AMT].

4.4.3 Hodnocení metodiky

***Metapopis metodiky* Enterprise Unified Process**		
položka	*význam, hodnoty číselníku*	
ID metodiky	MET4	
Název metodiky	Enterprise Unified Process	
Zkratka	EUP	
Autoři	Scott W. Ambler	
Rok vzniku	1999	
Zaměření	**EM**	globální metodika
Rozsah	**fáze**	IST, UST, GAN, DAN, IMP, ZAV, PUR
	dimenze	HW, TECH, DAT, FUN, UI
	role	všechny role definované v podkapitole **Chyba! Nenalezen zdroj odkazů.**
Váha metodiky	**HM**	těžká
Typ řešení	**NEW**	vývoj nového řešení (na zelené louce)
Doména	**CSW**	obecný SW
Přístup k řešení	**OO**	objektový vývoj ve všech fázích
	OR	objektový vývoj s relační databází
Slovní charakteristika	Enterprise Unified Process představuje rozšíření metodiky Rational Unified Process ve dvou směrech – rozšíření na úroveň celé organizace a rozšíření o provoz, údržbu a odstranění softwarového produktu.	
Poznámka		

Metodika EUP překonává omezení metodiky RUP. EUP je globální metodikou zaměřenou na budování informačního systému na úrovni celé organizace. Postihnuje tedy takové procesy jako je řízení portfolia projektů, řízení znovupoužitelnosti na úrovni celé organizace, vytváření globální architektury apod. Významné je zařazení fází pro provoz, údržbu a odstranění produktu. EUP ovšem stejně jako RUP zůstává zaměřen jen na malou množinu rolí a dimenzí (pouze role a dimenze úzce spjaté se softwarovým inženýrstvím) a pouze na objektově orientovaný vývoj nového řešení respektive rozvoj řešení.

5 Agilní metodiky

Změny technologií a ekonomického prostředí, ke kterým v současnosti dochází, a požadavky na rychlé zavedení IS/ICT vyžadují změny v metodikách. Tradiční rigorózní metodiky přestávají v takových podmínkách vyhovovat a začínají se prosazovat metodiky, které umožňují vytvořit řešení velmi rychle a pružně jej přizpůsobovat měnícím se požadavkům. Tyto metodiky jsou označovány jako agilní. Jedná se o různé metodiky, které vznikaly od druhé poloviny 90. let a které prosazují myšlenku, že jedinou cestou, jak prověřit správnost navrženého systému, je vyvinout jej (nebo jeho část) co nejrychleji, předložit zákazníkovi a na základě zpětné vazby jej upravit. Každá z agilních metodik je svým způsobem specifická, ale všechny jsou postaveny na stejných principech a hodnotách. Proto se sešli představitelé těchto přístupů v únoru 2001, podepsali „Manifest agilního vývoje software“ a vytvořili „Alianci pro agilní vývoj software“.

5.1 Hlavní principy agilních metodik

„Manifest agilního vývoje software“ deklaruje čtyři hodnoty, přičemž prvky na levé straně mají větší relativní význam než prvky na pravé straně. „Manifest“ uvádí (přeloženo dle [Fowler,Highsmith]):

„Odhalili jsme lepší způsob vývoje software, sami jej používáme a chceme pomoci i ostatním, aby jej používali. Z toho pohledu dáváme přednost:

- *individualitám a komunikaci* před procesy a nástroji,
- *provozuschopnému software* před obsažnou dokumentací,
- *spolupráci se zákazníkem* před sjednáváním kontraktu,
- *reakci na změnu* před plněním plánu.“

„Manifest“ tedy deklaruje zásadní principy, které je třeba při vývoji software uplatňovat. Na jejich základě bylo definováno 10 hlavních principů agilních metodik, které jsou v následujících odstavcích blíže charakterizovány (zpracováno dle [Fowler,Highsmith]):

1. **Nejvyšší prioritou je včas a kontinuálně dodávat software, který zákazníkům přináší hodnotu.**

   Zákazníka nezajímají dokumenty, diagramy nebo integrace stávajících systémů, ale zajímá jej, zda dostane fungující software v každé iteraci a zda poskytovaná funkcionalita uspokojí jeho potřeby. Tradiční přístupy předpokládají, že *splnění plánu = úspěch projektu = hodnota pro zákazníka*. V dnešní době je však nutné, aby hodnota pro zákazníka byla neustále prověřována, protože je předmětem změny.

2. **Změnu požadavků je možné provést i v pozdějších fázích vývoje, protože tím může zákazník získat konkurenční výhodu.**

   Agilní přístup se snaží realizovat změny efektivně a řídit rizika negativních dopadů. Je třeba realizovat krátké iterace (od dvou týdnů do dvou měsíců) a na jejich konci dodávat fungující

software. Iterativní vývoj je prosazován i tradičními metodikami, agilní přístupy však kladou důraz na zkrácení cyklu dodání přírůstku.

3. **Uživatelé a vývojáři spolupracují denně na projektu.**

   Agilní přístupy radikálně mění koncept specifikace požadavků. Východiskem je přesvědčení, že není možné dohodnout a podepsat požadavky na začátku projektu. Proto definují na začátku jen hrubé požadavky, které se potom na základě každodenních jednání s uživateli zpodrobňují a dokonce i mění. To zdůrazňuje spoluúčast uživatelů na definování požadavků a přenesení odpovědnosti za projekt na uživatele.

4. **Motivovaní jedinci, kteří mají vytvořeny podmínky pro práci a mají podporu vedení, jsou klíčovým faktorem úspěchu projektu.**

   I když nasadíme všechny možné prostředky a nástroje, jsou to nakonec lidé, kteří rozhodují o úspěchu či neúspěchu projektu. Lidé musí být vhodně motivováni a projekt musí být podporován všemi zainteresovanými.

5. **Nejefektivnějším způsobem přenosu informací v rámci vývojového týmu je osobní komunikace.**

   Agilním metodikám bývá vytýkána nedostatečná dokumentace. Dokumentace ale není cílem projektu. Smyslem dokumentace je pochopení problému, kterého se mnohem lépe, rychleji a s menšími náklady dosáhne používáním přímých technik komunikace.

6. **Primární mírou úspěchu je fungující software.**

   Ani plnění plánu, ani existující dokumentace návrhu nezajistí úspěch projektu. Je třeba mít fungující systém.

7. **Agilní procesy předpokládají „zdravý“ vývoj.**

   Vývojáři běžně pracují přesčas a o víkendech, což zákonitě snižuje jejich produktivitu. Je třeba vymezit pracovní prostor (cca 40 hodin týdně), v jehož rámci zůstane tým v dobré kondici.

8. **Perfektní technické řešení i návrh.**

   I když jsou agilní přístupy podobné rychlému vývoji aplikací (RAD Rapid Application Developmnent) v rychlosti a flexibilitě, velmi se liší v kvalitě návrhu a řešení. Agilní přístupy zdůrazňují kvalitu návrhu, která je nezbytná pro realizaci změn. Návrh není samostatnou etapou, která je plně dokončena před zahájením implementace, ale je to iterativní činnost prováděná v průběhu projektu.

9. **Zásadním požadavkem je jednoduchost řešení, tj. umění maximalizovat množství neudělané práce.**

   Při vývoji software můžeme použít množství postupů a metod. V agilních projektech je kladen důraz na jednoduché postupy.

10. **Nejlepší architektury, požadavky a návrhy vznikají ze samoorganizujících se týmů.**

    Tento princip zdůrazňuje kreativitu lidí, častou komunikaci a přizpůsobování metodiky.

Pod hlavičku agilních metodik můžeme zařadit následující metodiky a přístupy, které jsou v dalších podkapitolách stručně charakterizovány:

- *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*).

5.2 Dynamic Systems Development Method (DSDM)

5.2.1 Identifikace metodiky a zdrojů

Metodika *Dynamic Systems Development Method* (DSDM) vznikla ve Velké Británii v první polovině 90 let. O rozvoj metodiky a její rozšiřování se stará DSDM konsorcium (zdroj http://www.dsdm.org). Metodika DSDM má ze všech agilních metodik nejlépe propracovaný systém školení a kvalitní dokumentaci a je populární jak v Evropě, tak v USA. Popis metodiky je možné nalézt v [DSDM], [Highsmith,2000], [AgileMet].

5.2.2 Charakteristika metodiky

Metodika DSDM představuje rozšíření praktik rychlého vývoje aplikací (RAD). Její charakteristiky lze odvodit ze slov, která tvoří název metodiky. Slovo „dynamic“ reprezentuje schopnost přizpůsobit se změnám v průběhu procesu vývoje. Slovo „systems“ bychom dnes spíše nahradili slovem „solutions“ ve smyslu obchodního řešení. Písmeno „D“ původně reprezentující slovo „development“, bychom dnes mohli nahradit slovem „delivery“, které akcentuje význam dodaného řešení. Také poslední písmeno „M“, které znamená „method“, bychom dnes mohli nahradit slovem „model“. I tento posun výkladu zkratky ukazuje neustálý vývoj metodiky. DSDM je postaveno na 9 principech [DSDM]:

- aktivní zapojení uživatele,
- tým s rozhodovací pravomocí,
- časté dodávky produktů,
- klíčovým kritériem pro přijetí dodávky je podpora podnikových cílů,
- iterativní a inkrementální vývoj jako nástroj postupného přibližování k žádoucímu řešení,
- změny v průběhu vývoje,
- definice požadavků na hrubé úrovni,

- testování v průběhu celého životního cyklu,
- spolupráce mezi členy týmu.

Metodika DSDM rozděluje proces vývoje do třech hlavních fází – *Funkční model* (Functional Model), *Návrh* (Design and Build*)* a *Implementace* (Implementation*)*, kterým předchází *Studie proveditelnosti* (Feasibility Study) *a Byznys studie* (Business Study). Hlavní fáze probíhají iterativně. Obsahem fáze *Funkční model* je sběr a prototypování funkčních požadavků. Většina požadavků je dokumentována jako prototypy. Ve fázi *Návrh* jsou prototypy zpodrobňovány, tak aby podporovaly všechny požadavky (i nefunkční) a je navrhováno řešení. Náplní fáze *Implementace* je realizace navrženého řešení, zhodnocení projektu, školení uživatelů a další činnosti.

5.2.3 Hodnocení metodiky

Metapopis metodiky **Dynamic Systems Development Method**		
položka	*význam, hodnoty číselníku*	
ID metodiky	MET5	
Název metodiky	Dynamic Systems Development Method	
Zkratka	DSDM	
Autoři	DSDM konsorcium	
Rok vzniku	1995	
Zaměření	**PM**	projektová metodika
Rozsah	**fáze**	UST, GAN, DAN, IMP, ZAV
	dimenze	HW, TECH, DAT, FUN, UI
	role	vedoucí projektu, uživatel, hlavní programátor, analytik, vývojář, tester
Váha metodiky	**LM**	lehká
Typ řešení	**NEW**	vývoj nového řešení (na zelené louce)
Doména	**CSW**	obecný SW
Přístup k řešení	**RO**	RAD vývoj s objektovou analýzou
	OO	objektový vývoj ve všech fázích
	OR	objektový vývoj s relační databází
Slovní charakteristika	Metodika je rozšířením praktik rychlého vývoje aplikací (RAD) s důrazem na kvalitu řešení. Metodika je velmi dobře dokumentována a řízeným způsobem rozšiřována.	
Poznámka		

Metodika DSDM je projektovou metodikou, která je zaměřena zejména na softwarově inženýrskou oblast, méně se zabývá oblastí řízení. Je zaměřena pouze na vývoj nového řešení, kombinuje přístup rychlého vývoje aplikací (Rapid Application Development) s objektově orientovaným vývojem. Základní technikou používanou při analýze a návrhu je prototypování. Přínosem metodiky je řízení jejího rozvoje, propagace, školení a implementace.

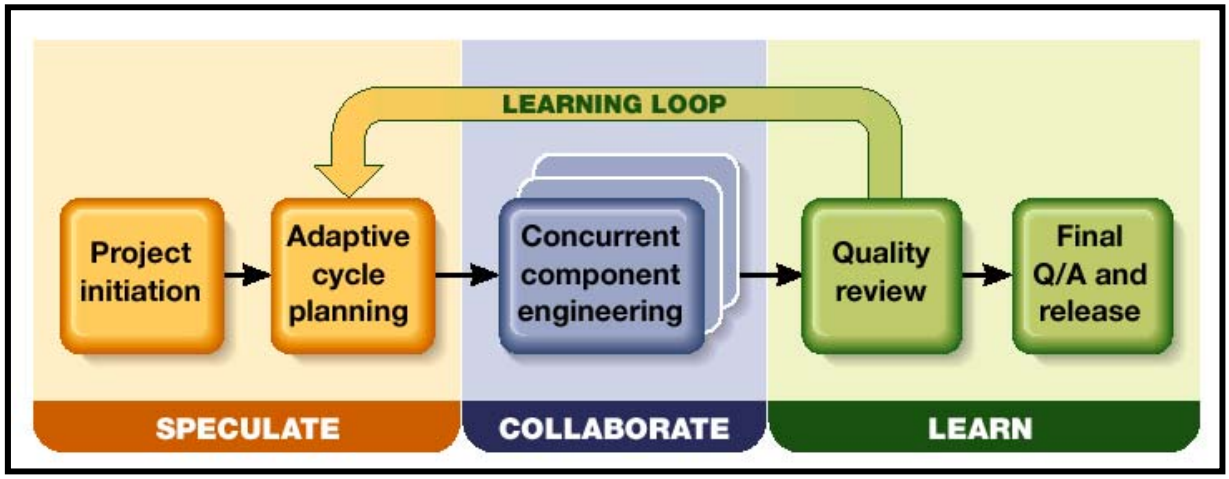
5.3 Adaptive Software Development (ASD)

5.3.1 Identifikace metodiky a zdrojů

Metodika *Adaptive Software Development* (ASD) představuje filosofické zázemí pro agilní metodiky. Autorem je Jim Highsmith. Popis metodiky je možné nalézt v [Highsmith,2000],[AgileMet].

5.3.2 Charakteristika metodiky

Metodika ASD je silně ovlivněna teorií komplexních adaptivních systémů. Změnám, které nastávají, se nesmíme bránit, ale musíme se na ně adaptovat. Proto je v metodice ASD statický životní cyklus „Plan–Design–Build“, typický pro tradiční metodiky, nahrazen dynamickým cyklem „Speculate–Collaborate–Learn“. Základem tohoto dynamického životního cyklu je kontinuální učení a hybnou silou jsou neustálé změny. V klasickém životním cyklu je problematická fáze *Plánování*. Plán totiž ze své podstaty brání inovacím a produkuje výsledky, které jsou sice plánovány, ale nejsou potřebné. *Spekulace* dává mnohem více prostoru pro změny, přiznává nejistotu, podporuje zkoumání a experimentování. Při klasickém postupu jsou odchylky od plánu chápány většinou jako chyby, ASD v nich vidí příležitosti k učení. Druhou podstatnou součástí adaptivního životního cyklu je *Spolupráce* (Collaboration). Pro vytváření současných aplikací je třeba sebrat velké množství informací, analyzovat je a aplikovat na řešení problému. Jde o mnohem větší objem informací než je schopen jedinec zvládnout, a proto je rozhodující schopnost spolupracovat. Týmy musí spolupracovat na řešení technických problémů i věcných požadavků. Rozhodování závisí na třetí složce adaptivního životního cyklu – učení. Musíme své znalosti neustále prověřovat, učit se z minulých chyb i úspěchů.



obrázek 5.1: Základní adaptivní životní cyklus zdroj [Highsmith,7/2000]

Životní cyklus metodiky ASD zachycuje obrázek 5.1. Cyklus se skládá ze tří fází *Spekulace, Spolupráce* a *Učení*. Předmětem fáze *Spekulace* jsou činnosti spojené se zahájením projektu, určení termínu ukončení projektu, optimálního počtu iterací, náplně každé iterace. V rámci fáze *Spolupráce* probíhá paralelně vývoj komponent a výsledkem jsou fungující komponenty. V rámci fáze *Učení* je

třeba zhodnotit kvalitu řešení z pohledu zákazníka i z hlediska technologického, fungování týmu, používané praktiky a stav projektu.

5.3.3 Hodnocení metodiky

***Metapopis metodiky* Adaptive Software Development**		
položka	*význam, hodnoty číselníku*	
ID metodiky	MET6	
Název metodiky	Adaptive Software Development	
Zkratka	ASD	
Autoři	Jim Highsmith	
Rok vzniku	1997	
Zaměření	**PM**	projektová metodika
Rozsah	**fáze**	UST, GAN, DAN, IMP, ZAV
	dimenze	HW, TECH, DAT, FUN, UI
	role	vedoucí projektu, uživatel, hlavní programátor analytik, vývojář, tester
Váha metodiky	**LM**	lehká
Typ řešení	**NEW**	vývoj nového řešení (na zelené louce)
Doména	**CSW**	obecný SW
Přístup k řešení	**RO**	RAD vývoj s objektovou analýzou
	OO	objektový vývoj ve všech fázích
	OR	objektový vývoj s relační databází
Slovní charakteristika	ASD představuje filosofické zázemí pro agilní metodiky. ASD je založena na dynamickém životním cyklu „Speculate– Collaborate– Learn“.	
Poznámka		

Metodika ASD je určena pro projekty charakterizované vysokou rychlostí, změnami a neurčitostí [Highsmith,7/2000]. Je to „lehká“ metodika, která se zabývá nejen oblastí softwarově inženýrskou, ale i oblastí řízení. Přináší nový přístup k řízení nazvaný „Leadership – Collaboration management“. ASD je projektovou metodikou zaměřenou na objektově orientovaný a komponentový vývoj nového řešení. Přínosem metodiky je důraz na fázi učení.

5.4 Lean development

5.4.1 Identifikace metodiky a zdrojů

Metodika *Lean development*, jejímž autorem je Robert Charette, je aplikací principů známých jako *Lean Manufacturing* a *Total Quality Management* na oblast vývoje software. Popis metodiky je možné nalézt v [Charette,2002], [Highsmith,2000], [AgileMet].

5.4.2 Charakteristika metodiky

Metodika *Lean Development* je inspirována postupy, které byly uplatňovány ve výrobě zejména v 80. letech (štíhlá výroba - lean production). Metodika je založena na konceptu **dynamické stability.** Schopnost přizpůsobit se rychle a efektivně požadavkům (dynamická část) je spojena se schopností

vytvářet stabilní, neustále se zlepšující vnitřní procesy, které mají obecnou platnost a přizpůsobují se širokému okruhu produktů. Cílem *Lean development* je vytváření software tolerantního ke změnám s třetinovou lidskou prací, s třetinovým časem, s třetinou investic do nástrojů a metod, s třetinovou námahou přizpůsobit se novému tržnímu prostředí. Následující přehled ukazuje aplikaci 10 pravidel štíhlé výroby na oblast vývoje software:

Pravidlo 1: Odstranit zbytečné.

Aplikace tohoto pravidla znamená odstranit vše, co nepřináší hodnotu konečnému produktu. Dokumenty, diagramy a modely vytvářené při vývoji software pohlcují zdroje, ale nejsou nutnou součástí finálního produktu.

Pravidlo 2: Minimalizovat zásoby (minimalizovat artefakty).

Zásobou při vývoji software je dokumentace, která není součástí finálního produktu. Nejlepší přístup, jak minimalizovat dokumentaci, je dosáhnout určité úrovně abstrakce v dokumentaci. Místo 100 stran detailní specifikace stačí 10 stran pravidel a dokumentace odchylek.

Pravidlo 3: Maximalizovat tok (zkrátit čas vývoje).

Jestliže je třeba zkrátit dobu vývoje, je třeba redukovat práci na procesu. Iterativní vývoj je aplikací tohoto principu.

Pravidlo 4: Vývoj tažený poptávkou (rozhodovat co nejpozději).

Praktiky vývoje software, které dokáží přizpůsobit dodávku software požadavkům uživatelů, představují v měnícím se prostředí konkurenční výhodu. Uživatelé nejsou schopni definovat současné potřeby, natož potřeby budoucí. Pokud je návrh zařazen na začátku životního cyklu, s největší pravděpodobností se dostane do rozporu s požadavky.

Pravidlo 5: Pracovníci s rozhodovací pravomocí (rozhodovat co nejníže).

Vývojáři musí chápat, jak jejich práce přispívá celkovému cíli, musí vědět, co mají vykonat a do kdy, a musí mít možnost rozhodovat.

Pravidlo 6: Uspokojovat požadavky zákazníků (nyní i v budoucnu).

Nejčastější důvody neúspěchu projektů byly způsobeny chybějícími, nekompletními nebo nesprávnými požadavky. Metodiky vývoje software na to odpověděly praktikami detailní specifikace uživatelských požadavků, které uživatel odsouhlasí. Uživatel ale není schopen předem dohlédnout všechny potřeby. Úplná specifikace požadavků trvá dlouho, a tak se zvyšuje riziko, že zjištěné požadavky nebudou odpovídat skutečným potřebám.

Pravidlo 7: Zavést zpětnou vazbu.

Lean development vychází z toho, že pokud není možné definovat detailně všechny požadavky předem, je třeba zavést zpětnou vazbu a doplňovat je postupně. To ale znamená provádět změny v průběhu vývoje.

Pravidlo 8: Odstranit lokální optimalizaci.

V době neustálých změn nemá smysl optimalizovat stávající řešení.

Pravidlo 9: Partnerství s dodavateli.

Důležitá je hodnota pro zákazníka, pro urychlení a zlepšení kvality je možné nakupovat komponenty.

Pravidlo 10: Zavést kulturu pro neustálé zlepšování

Vytvořit podmínky a motivaci zlepšovat procesy při vývoji software.

5.4.3 Hodnocení metodiky

***Metapopis metodiky* Lean development**		
položka	*význam, hodnoty číselníku*	
ID metodiky	MET7	
Název metodiky	Lean development	
Zkratka		
Autoři	Robert Charette	
Rok vzniku	2001	
Zaměření	**PM**	projektová metodika
Rozsah	**fáze**	UST, GAN, DAN, IMP, ZAV
	dimenze	HW, TECH, DAT, FUN, UI
	role	vedoucí projektu, uživatel, hl.programátor, analytik, vývojář, tester
Váha metodiky	**LM**	lehká
Typ řešení	**NEW**	vývoj nového řešení (na zelené louce)
Doména	**CSW**	obecný SW
Přístup k řešení	**RO**	RAD vývoj s objektovou analýzou
	OO	objektový vývoj ve všech fázích
	OR	objektový vývoj s relační databází
Slovní charakteristika	Cílem Lean development je vytváření software tolerantního ke změnám s třetinovou lidskou prací, s třetinovým časem, s třetinou investic do nástrojů a metod, s třetinovou námahou přizpůsobit se novému tržnímu prostředí. Metodika je zaměřena zejména na řízení vývoje software a na řízení rizik.	
Poznámka		

Zatímco většina agilních metodik se zabývá taktickou úrovní, *Lean Development* se zaměřuje spíše na strategickou úroveň s vazbou na podnikovou strategii. *Lean Development* je nástrojem přechodu na podnikání tolerantní ke změnám (change tolerant bussiness) a „risk entrepreneurship“. Podobně jako metodika *Scrum* je *Lean Development* zaměřen zejména na řízení vývoje software, méně pak na softwarově inženýrskou oblast.

5.5 Feature-Driven Development (FDD)

5.5.1 Identifikace metodiky a zdrojů

Metodika *Feature–Driven Development* (FDD), jejímiž autory jsou Jeff De Luca a Peter Coad, je agilní metodika, která zachovává procesní řízení a zdůrazňuje úlohu modelování při vývoji. Popis metodiky je možné nalézt v [FDD], [Highsmith,2000], [AgileMet].

5.5.2 Charakteristika metodiky

Metodika FDD je založena na iterativním vývoji, který je řízen užitnými vlastnostmi produktu (feature–driven). Vývoj začíná vytvořením celkového modelu a pokračuje posloupností dvoutýdenních iterací, ve kterých se provádí návrh i realizace pro jednotlivé užitné vlastnosti. **Užitná vlastnost** (feature) je malý výsledek užitečný z pohledu zákazníka, který je srozumitelný, měřitelný a realizovatelný spolu s dalšími v rámci dvoutýdenní iterace. FDD se skládá z 5 procesů:

- vytvoření celkového objektového modelu (Develop an Overall Model),
- sestavení seznamu užitných vlastností (Build a Features List),
- plánování pro užitnou vlastnost (Plan by Feature),
- návrh pro užitnou vlastnost (Design by Feature),
- realizace pro užitnou vlastnost (Build by Feature).

FDD stejně jako ostatní agilní metodiky nepřeceňuje význam procesů při vývoji. Cílem je fungující produkt, nikoli splnění předepsaného procesu. Přesto definuje „lehké“ procesy, které jsou popsány cca na 2 stránkách textu a které umožňují škálovat metodiku na větší projekty, rychleji zapojit nové zaměstnance, stanovit priority, zaměřit se na přínosy.

5.5.3 Hodnocení metodiky

***Metapopis metodiky* Feature–Driven Development**		
položka	*význam, hodnoty číselníku*	
ID metodiky	MET8	
Název metodiky	Feature–Driven Development	
Zkratka	FDD	
Autoři	Jeff De Luca , Peter Coad	
Rok vzniku	1998	
Zaměření	**PM**	projektová metodika
Rozsah	**fáze**	UST, GAN, DAN, IMP, ZAV
	dimenze	HW, TECH, DAT, FUN, UI
	role	vedoucí projektu, uživatel, hlavní programátor, vlastník třídy analytik, vývojář, tester
Váha metodiky	**MM**	střední
Typ řešení	**NEW**	vývoj nového řešení (na zelené louce)
Doména	**CSW**	obecný SW
Přístup k řešení	**OO**	objektový vývoj ve všech fázích
	OR	objektový vývoj s relační databází
Slovní charakteristika	Metodika FDD je založena na iterativním vývoji, který je řízen užitnými vlastnostmi produktu. Vývoj začíná vytvořením celkového modelu a pokračuje posloupností dvoutýdenních iterací, ve kterých se provádí návrh i realizace pro jednotlivé užitné vlastnosti. **Užitná vlastnost** (*feature*) je malý výsledek užitečný z pohledu zákazníka.	
Poznámka		

Podobně jako ostatní agilní metodiky je FDD projektová metodika zaměřená na objektově orientovaný vývoj nového software. Na rozdíl od ostatních agilních metodik však podporuje procesy, byť poměrně

„lehké“, modelování (vytvoření celkového modelu) a používání CASE nástrojů. Z toho důvodu ji můžeme zařadit mezi středně „těžké“ metodiky.

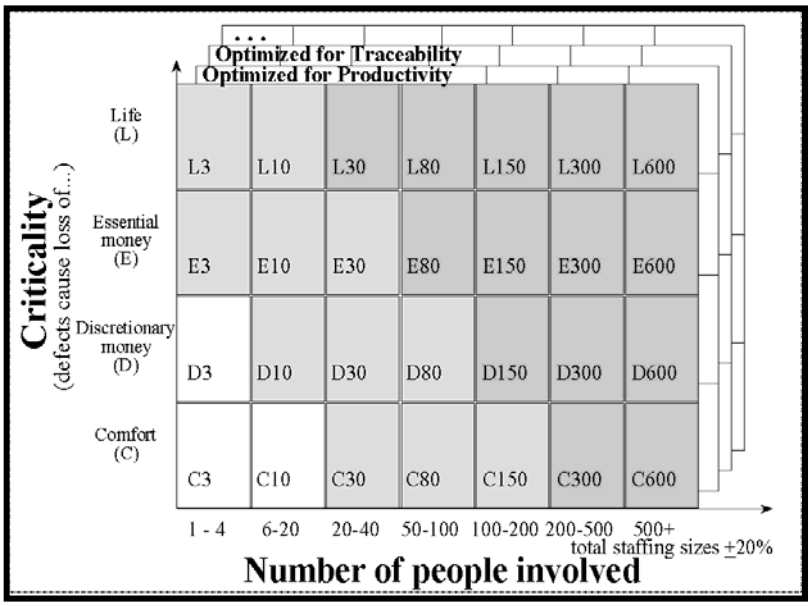
5.6 Crystal metodiky

5.6.1 Identifikace metodiky a zdrojů

Crystal je rodina metodik, které jsou určeny pro různé typy projektů. Autorem rodiny metodik je Alistair Cockburn. Popis metodiky je možné nalézt v [Cockburn, MetPerProj], [Cockburn,Cryst], [Highsmith,2000], [AgileMet].

5.6.2 Charakteristika metodiky

Jádrem všech metodik rodiny *Crystal* je síla komunikace a lehkost produktu. Prvky metodiky se přizpůsobují pro každý projekt. Obrázek 5.2 zachycuje schematicky jednotlivé metodiky. Výběr vhodné metodiky z rodiny se provádí na základě velikosti projektu, kterou určuje počet členů týmu (osa x), a důležitosti systému (osa y). Třetí rozměr určuje hledisko, pro které je metodika optimalizována (produktivita, trasovatelnost apod.). Jednotlivé metodiky jsou pojmenovány podle barev, „nejlehčí“ metodika je nazvána *Clear*, potom následuje *Yellow, Orange, Red, Maroon, Blue, Violet* atd. Například *Orange* je *D40* metodika, to znamená, že je určena pro týmy do 40 lidí, kteří sedí v jedné budově a pracují na projektu, který může znamenat větší ztrátu peněz.



obrázek 5.2: Rodina metodik Crystal zdroj [CockCryst]

5.6.3 Hodnocení metodiky

***Metapopis metodiky* Crystal**	
položka	*význam, hodnoty číselníku*
ID metodiky	MET9
Název metodiky	Rodina metodik Crystal

Metapopis metodiky **Crystal**		
Zkratka	ASD	
Autoři	Alistair Cockburn	
Rok vzniku	1998	
Zaměření	**PM**	projektová metodika
Rozsah	**fáze**	UST, GAN, DAN, IMP, ZAV
	dimenze	HW, TECH, DAT, FUN, UI
	role	vedoucí projektu, uživatel, hlavní programátor analytik, vývojář, tester
Váha metodiky	**LM**	lehká
Typ řešení	**NEW**	vývoj nového řešení (na zelené louce)
Doména	**CSW**	obecný SW
Přístup k řešení	**RO**	RAD vývoj s objektovou analýzou
	OO	objektový vývoj ve všech fázích
	OR	objektový vývoj s relační databází
Slovní charakteristika	Crystal představuje skupinu metodik pro různé druhy projektů, které se liší důležitostí, velikostí týmu a tím, na co je projekt optimalizován.Všechny metodiky mají společné hodnoty a principy, liší se použitými technikami, rolemi, nástroji a standardy.	
Poznámka		

Crystal představuje skupinu metodik pro různé druhy projektů, které se liší důležitostí systému, velikostí týmu kritériem optimalizace. Všechny metodiky patří do kategorie projektových metodik a jsou opět zaměřeny na objektově orientovaný vývoj nového řešení. Přínosem rodiny metodik je princip škálování podle typu projektu a důraz na lidský faktor.

5.7 Scrum

5.7.1 Identifikace metodiky a zdrojů

Autory metodiky *Scrum* jsou Ken Schwaber, Jeff Sutherland a Mike Beedle. Popis metodiky je možné nalézt v [Scrum2], [Highsmith,2000], [AgileMet].

5.7.2 Charakteristika metodiky

Přístup *Scrum* je založen na přesvědčení, že vývoj software není definovaný proces, jak rigorózní metodiky předpokládají, ale empirický proces, a proto vyžaduje úplně odlišný styl řízení. Název Scrum byl vybrán podle skrumáže (mlýna) v rugby proto, aby zdůraznil, že metodika *Scrum* je podobně jako hra rugby adaptivní, rychlá a samoorganizující. Metodika *Scrum* je zaměřena především na řízení projektu. Vývoj software probíhá v rámci 30 denních iterací nazývaných *Sprint*, na jejichž konci je dodána vybraná množina užitných vlastností. Klíčovou praktikou metodiky *Scrum* je používání každodenních 15 minutových porad (*Scrum Meetings)*, které slouží pro koordinaci a integraci prací. Jak ukazuje tabulka 5.1, metodika *Scrum* definuje 4 fáze životního cyklu.

fáze	***popis***
plánovací fáze *(planning phase)*	specifikují se první požadavky, plán dodávek a architektonická a byznys vize

vynášecí fáze *(staging phase)*	do souboru požadavků (backlog) jsou připojeny nefunkční požadavky
fáze vývoje *(development phase)*	jeden nebo více týmů dodává funkcionalitu s nejvyšší prioritou každých 30 dní, na konci každé iterace tým předvede výsledek
fáze dodávky *(release phase)*	předání produktu uživatelům

tabulka 5.1: Fáze metodiky Scrum

První a poslední fáze obsahuje definované procesy, u kterých jsou určeny vstupy, výstupy a procesy transformace vstupů na výstupy. Tyto procesy vyjadřují explicitní znalost a jejich provádění je lineární. Fáze vývoje (*Sprint*) je naproti tomu empirický proces, jehož činnosti nelze zpravidla identifikovat a řídit. Tato fáze vystupuje jako černá skříňka, která vyžaduje vnější řízení. Pravidla pro 30 denní *Sprint* jsou jednoduchá. Každý člen týmu má přidělen úkol, pracuje na splnění cíle *Sprintu* a účastní se denních porad (*Scrum meetings*). Tyto porady mají informační charakter a jsou velmi efektivně řízeny. Umožňují monitorovat stav projektu, zaměřit se na to, co je třeba udělat pro úspěch projektu a jak řešit problémy. Denní porady se konají ve stejný čas na stejném místě a trvají maximálně 30 minut (optimálně 15 minut). Porady řídí *Scrum Master*, účastní se jich všichni členové týmu a navštěvují je také manažeři. Denní porady umožňují týmu sdílet znalosti. Na poradách musí každý účastník zodpovědět tři otázky:

- které položky dokončil od minulé porady,
- které nové úkoly má řešit,
- jaká vidí omezení a překážky pro řešení úkolů.

5.7.3 Hodnocení metodiky

***Metapopis metodiky* Scrum**		
položka	*význam, hodnoty číselníku*	
ID metodiky	MET10	
Název metodiky	Scrum	
Zkratka		
Autoři	Ken Schwaber, Jeff Sutherland, Mike Beedle	
Rok vzniku	1995	
Zaměření	**PM**	projektová metodika
Rozsah	**fáze**	UST, GAN, DAN, IMP, ZAV
	dimenze	
	role	vedoucí projektu, uživatel, vývojář
Váha metodiky	**LM**	lehká
Typ řešení	**NEW**	vývoj nového řešení (na zelené louce)
Doména	**CSW**	obecný SW
Přístup k řešení	**RO**	RAD vývoj s objektovou analýzou
	OO	objektový vývoj ve všech fázích
	OR	objektový vývoj s relační databází
Slovní charakteristika	Metodika Scrum je zaměřena hlavně na oblast řízení projektu. Vývoj probíhá v 30 denních iteracích nazývaných Sprint, ve kterých se dodává vybraná množina užitných	

Metapopis metodiky **Scrum**	
	vlastností. Klíčovou praktikou je používání každodenních 15 minutových Scrum Meetings pro koordinaci a integraci prací.
Poznámka	

Scrum je metodika kategorie projektových metodik zaměřená především na řízení projektu. Chápe procesy při vývoji software jako empirické procesy, které není možné předvídat, ale je nutné je monitorovat. K tomu poskytuje praktiky jako denní porady, monitorování *Sprintu* (30 denní iterace) pomocí *Backlog graph* a další. Metodika *Scrum* explicitně snižuje chaos při vývoji tím, že stabilizuje úkoly pro 30 denní *Sprint*. Metodika *Scrum* je popsána jako jazyk vzorů (pattern language).

5.8 Extrémní programování (XP)

5.8.1 Identifikace metodiky a zdrojů

Extrémní programování je metodika určená zejména pro malé až středně velké týmy (2 – 10 programátorů), které vyvíjejí software, jehož zadání není jasné a nebo se mění. Autory metodiky jsou Kent Beck, Ward Cunningham a Ron Jeffries. Popis metodiky je možné nalézt v [Beck,2002], [Highsmith,2000], [AgileMet].

5.8.2 Charakteristika metodiky

Extrémní programování vychází z principů a postupů běžných při vývoji software, které však dovádí do extrémů:

- pokud se osvědčují revize zdrojového textu, budeme kód neustále revidovat (**párové programování**),
- jestliže se osvědčuje testování, budou všichni vývojáři neustále testovat (**testování jednotek**) a testovat budou také zákazníci (**testování funkcionality**),
- osvědčuje-li se návrh, zařadíme jej jako každodenní činnost (**refaktorizace**),
- pokud se osvědčuje jednoduchost, vždy zvolíme nejjednodušší řešení, které vyhovuje požadovaným funkcím (**to nejjednodušší, co ještě může fungovat**),
- jsme-li přesvědčeni o důležitosti architektury, budou ji všichni neustále definovat a rozpracovávat (**metafora**),
- jestliže jsme přesvědčeni o důležitosti testování integrace, budeme integrovat a testovat několikrát denně (**nepřetržitá integrace**),
- pokud se osvědčují krátké iterace, uděláme je opravdu krátké: vteřiny, minuty a hodiny nikoli týdny, měsíce a roky (**plánovací hra**).

Extrémní programování zdůrazňuje hodnotu společného, jednoduchost, odezvu a odvahu. Důležitými aspekty XP jsou nový pohled na náklady změny, důraz na kvalitu technického řešení jako výsledku refaktorizace a vytváření testů před kódováním.

5.8.3 Hodnocení metodiky

***Metapopis metodiky* Extrémní programování**		
položka	*význam, hodnoty číselníku*	
ID metodiky	MET11	
Název metodiky	Extrémní programování	
Zkratka	XP	
Autoři	Kent Beck, Ward Cunningham, Ron Jeffries	
Rok vzniku		
Zaměření	**PM**	projektová metodika
Rozsah	**fáze**	UST, GAN, DAN, IMP, ZAV
	dimenze	HW, TECH, DAT, FUN, UI
	role	vedoucí projektu, uživatel, vývojář, tester
Váha metodiky	**LM**	lehká
Typ řešení	**NEW**	vývoj nového řešení (na zelené louce)
Doména	**CSW**	obecný SW
Přístup k řešení	**OO**	objektový vývoj ve všech fázích
	OR	objektový vývoj s relační databází
Slovní charakteristika	XP je metodika pro malé až středně velké týmy (2 – 10 programátorů), které vyvíjejí software a musí se vyrovnat se zadáním, které se rychle mění nebo není jasné. XP zdůrazňuje hodnotu společného, jednoduchost, odezvu a odvahu. Důležitými aspekty XP jsou nový pohled na náklady změny, důraz na kvalitu technického řešení jako výsledku refaktorizace a vytváření testů před kódováním.	
Poznámka		

Extrémní programování je velmi „lehká“ metodika, která zavádí specifické praktiky jako párové programování, refaktorizace, testy před kódováním a další. Principy a praktiky se týkají jak oblasti softwarově inženýrské, tak i řízení a organizace. I když je *Extrémní programování* známé jako „lehká“ metodika, přesto jde o velmi disciplinovaný proces. Ten však není podrobně popisován, ale je realizován velmi kvalifikovanými a disciplinovanými vývojáři za podpory vývojových nástrojů (pro refaktorizaci a testování).

5.9 Agilní modelování

5.9.1 Identifikace metodiky a zdrojů

Agilní modelování (dříve nazývané *Extrémní modelování,* XM) je metodika založená na praktikách, principech a hodnotách, které jsou odvozeny z hodnot metodiky *Extrémní programování*. Není to ucelená metodika, ale je to sada principů a praktik věnovaných modelování jako nedílné součásti vývoje software. *Agilní modelování* tak může být začleněno jak do tradičních metodik (například RUP), tak do agilních metodik (například *Scrum*, XP, apod.) Autorem je Scott Ambler. Popis metodiky je možné nalézt v [Ambler,8/2001SD], [Ambler,4/2001SD], [Ambler,AMWP].

5.9.2 Charakteristika metodiky

Modelování je základní součástí vývoje software jak v tradičních metodikách, tak v agilních metodikách. Modelování je prosazováno standardizačními organizacemi jako OMG a IEEE, výrobci CASE nástrojů a vývojových nástrojů a je součástí většiny metodik. V praxi se ale často neprovádí. Příčinou mohou být tzv. „mýty o modelování“, jak je formuloval Scott Ambler v [Ambler,8/2001SD]. Velmi častým a dosti nebezpečným mýtem je představa, že modelování je totožné s dokumentací. Vývojáři nechtějí ztrácet čas tvorbou dokumentace, a tak nemodelují, což potom vede k nízké kvalitě vytvářených systémů. Náčrtky na papíře, obrázky na tabuli, CRC[7] karty, nákresy uživatelského rozhraní nejsou dokumenty, ale přesto představují hodnotné modely. Modelování plní podobnou úlohu jako plánování. Při plánování není hodnotou plán samotný, ale proces plánování. Stejně tak při modelování je důležitý proces modelování. Druhý mýtus přeceňuje význam modelů vytvořených předem. Jeho zastánci věří, že je možné namodelovat vše předem a správně. Výsledkem je potom obrovské množství dokumentace místo fungujícího software. Tento mýtus je spojen s běžnou praxí zmrazení požadavků na začátku životního cyklu. To vede k tomu, že dodané systémy neodpovídají skutečným potřebám uživatelů. Dalším mýtem při modelování je představa, že je nutné používat CASE nástroj, který nejlépe zvládne složité modely. Mnohdy je ale účinnější vytvářet jednoduché modely, které zachycují jen důležité informace místo nepodstatných detailů. O překonání výše uvedených mýtů se snaží *Agilní modelování.* Principy agilního modelování uvádí tabulka 5.2 a praktiky agilního modelování tabulka 5.3 [Ambler,8/2001SD], [Ambler,4/2001SD], [Ambler,AMWP].

Principy agilního modelování	
princip	*vysvětlení*
Nejdůležitějším úkolem je vytvořit fungující software	cílem není vytvářet modely, ale vytvořit kvalitní software, který odpovídá potřebám zákazníka
Druhým nejdůležitějším úkolem je umožnit další práci na projektu	vytvořený software musí být natolik robustní, aby jej bylo možné dále rozvíjet, současně je třeba mít k dispozici takovou dokumentaci, aby byl další rozvoj možný
Obsah je mnohem důležitější než reprezentace	každý model může být reprezentován různými způsoby, je třeba vybrat dostačující způsob – aby splnil účel modelování, modely nemusí být tedy perfektní ani nemusí být nutně vytvořené v CASE nástrojích
Jednoduchost	nejjednodušší řešení je nejlepší řešení
Komunikace	hlavním smyslem modelování je komunikace mezi členy týmu navzájem i se zákazníky a zájmovými skupinami
Modelovat za určitým účelem	modely by se měly vytvářet jen když je jasné, pro koho je model určen a proč
Uchopit změnu	změny nastávají, je třeba je akceptovat
Změny je třeba dělat přírůstkově	je lepší měnit systém po malých částech než provést jednu všezahrnující změnu
Je třeba se učit od druhých	nikdo nemůže znát vše, je třeba se učit od druhých a rozvíjet své znalosti

[7] Class Responsibility Collaboration - metoda pro zjišťování odpovědností tříd.

Principy agilního modelování	
princip	*vysvětlení*
Znát své modely	existují různé modely, které odrážejí pohledy na systém, je třeba znát jejich silné a slabé stránky a efektivně je využívat
Přizpůsobení metodiky	metodiku je třeba přizpůsobit podle konkrétních podmínek
Maximalizovat výnosy z investic	je třeba maximálně zhodnotit vynaložené prostředky
Více modelů	k dispozici je široké spektrum modelů , je třeba použít nejvhodnější
Otevřená komunikace	lidé se musí cítit volní, díky otevřené komunikaci mohou dělat lepší rozhodnutí
Důraz na kvalitu	je třeba se zaměřovat na kvalitu software v celém procesu jeho vývoje, kvalitní software může splnit požadavky zákazníka, je snazší jej měnit
Rychlá zpětná vazba	nejcennější při vývoji je rychlá zpětná vazba, prostředkem pro její zajištění je – krátké iterace, prototypy, uživatel součástí týmu
Cestovat nalehko	tento princip vychází z paralely mezi vývojem software a slézáním hory, je třeba vytvářet jen tolik modelů a dokumentace, aby je člověk mohl vzít s sebou
Řídit se instinkty lidí	lidé se snaží dělat vše pro dobro věci, je dobré využít jejich tacit znalostí

tabulka 5.2: Principy agilního modelování

Praktiky agilního modelování	
praktika	*vysvětlení*
Aktivní účast investorů	úspěch projektu často závisí na aktivní účasti investorů – vedení podniku, operativní pracovníci, jiné týmy
Používání standardů při modelování	je třeba používat obecné standardy modelování
Využívání vzorů	je třeba používat architektonické, návrhové a analytické vzory
Používání správných artefaktů	artefaktů je velké množství a je třeba vybrat vhodný pro daný účel
Kolektivní vlastnictví	odvozena od praktiky extrémního programování umožňuje každému pracovat na libovolném modelu
Testovat modely	testování je základní prostředek vytváření kvalitního software, je třeba testovat i modely
Paralelní vytváření různých modelů	modely reprezentují různé pohledy na vytvářený systém, je třeba znát silné a slabé stránky jednotlivých typů modelů a vytvářet více modelů
Jednoduchý obsah	je třeba se snažit o jednoduchost obsahu modelů
Jednoduché zobrazení modelů	je třeba se snažit o jednoduchost zobrazení modelů, je vhodné používat podmnožinu notace, cílem je jednoduchý model, který zachycuje klíčové rysy
Odstranění dočasných modelů	většina vytvářených modelů je dočasná, když splní svůj účel, je třeba je zrušit
Veřejné vystavení modelů	podporuje princip otevřené komunikace

Praktiky agilního modelování	
praktika	*vysvětlení*
Formalizace požadovaných modelů	některé modely jsou požadovány např. vedením, ty je třeba formálně upravit
Přechod na jiné artefakty	protože modely reprezentují různé pohledy na vytvářený systém a vzájemně se doplňují, je třeba přecházet mezi modely
Modelování v malých přírůstcích	souvisí s přírůstkovým vývojem, není možné vytvořit detailní model předem, ale vytváří se postupně
Modelování pro komunikaci	smyslem modelování je komunikace v týmu, se zákazníkem, s vedením
Modelování pro pochopení	smyslem modelování je pochopení problému
Je nebezpečné modelovat sám	modelování je činnost, která vyžaduje abstrakci, zkušenosti, existuje více možností, jak model navrhnout, proto je vhodné modelovat s jinými
Testování modelů kódem	model je abstrakce, pro zjištění, zda bude skutečně fungovat, je třeba napsat kód
Znovupoužití existujících zdrojů	je třeba využívat hotové modely, vzory
Úprava jen, když je to nezbytné	úpravy modelů jsou náročné a měly by se realizovat, jen když je to nezbytně nutné
Používání nejjednodušších nástrojů	většina modelů může být malována na tabuli

tabulka 5.3: Praktiky agilního modelování

5.9.3 Hodnocení metodiky

***Metapopis metodiky* Agilní modelování**		
položka	*význam, hodnoty číselníku*	
ID metodiky	MET12	
Název metodiky	Agilní modelování	
Zkratka	AM	
Autoři	Scott W.Ambler	
Rok vzniku		
Zaměření	**PM**	projektová metodika
Rozsah	**fáze**	GAN, DAN
	dimenze	TECH, DAT, FUN, UI
	role	uživatel, analytik, vývojář
Váha metodiky	**LM**	lehká
Typ řešení	**NEW**	vývoj nového řešení (na zelené louce)
Doména	**CSW**	obecný SW
Přístup k řešení	**RO**	RAD vývoj s objektovou analýzou
	OO	objektový vývoj ve všech fázích
	OR	objektový vývoj s relační databází
Slovní charakteristika	Agilní modelování je založené na praktikách, principech a hodnotách, které jsou odvozeny z hodnot extrémního programování.	
Poznámka		

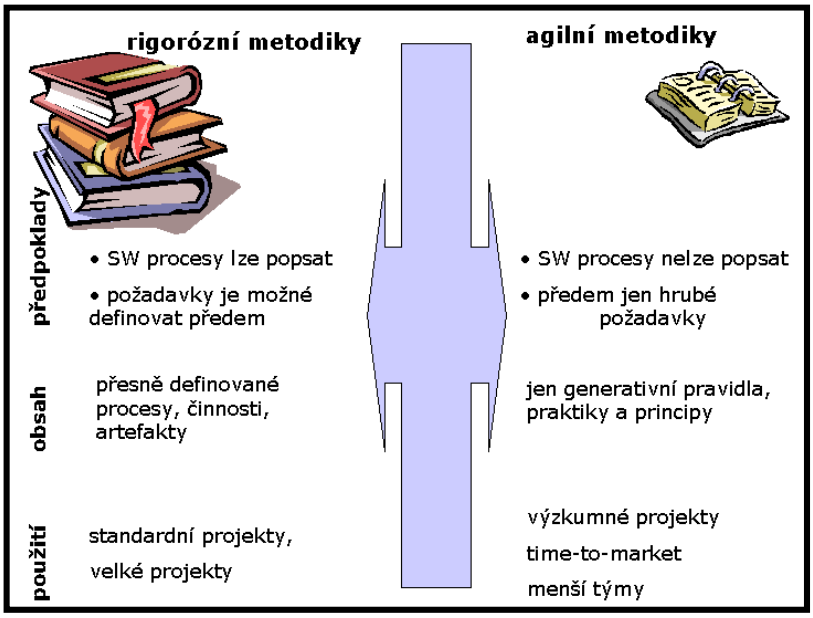
Agilní modelování je „lehká“ metodika, která vychází z prověřených modelovacích technik, ale přizpůsobuje je agilním přístupům. Přínosem metodiky *Agilní modelování* nejsou techniky modelování

jako takové, ale to, jakým způsobem jsou aplikovány. Agilní model je takový model, který plní svůj účel, je pochopitelný, je dostatečně přesný, je dostatečně konzistentní, je dostatečně detailní, přináší kladnou hodnotu a je co možná nejjednodušší.

5.10 Porovnání rigorózních a agilních metodik

Agilní metodiky představují ve své podstatě reengineering procesů při budování IS/ICT. Když byl v 90. letech prosazován reengineering v podnikových procesech, dostala jedna osoba (vlastník procesu) plnou kontrolu nad celým procesem. Podobně agilní přístup k vývoji software dává jednomu vývojáři plnou kontrolu nad všemi fázemi procesu vývoje – od přímé komunikace se zákazníkem při sběru požadavků až k realizaci. Agilní metodiky jsou metodiky projektové (dle kategorizace, kterou zavádí kapitola 3) a jsou zaměřeny na vývoj nového řešení (nikoli na údržbu a provoz).

Rigorózní a agilní metodiky představují dvě skupiny metodik, které vycházejí z odlišných předpokladů a odlišného pohledu na vývoj software. Výsledkem je jiný obsah a zaměření každé kategorie metodik a jiný okruh projektů, na které je vhodné tyto metodiky aplikovat (obrázek 5.3).



obrázek 5.3: Srovnání rigorózních a agilních metodik

Podrobné srovnání rigorózních a agilních metodik na základě různých hledisek poskytuje tabulka 5.4. I když jsou východiska, obsah, přístupy i použití rigorózních a agilních metodik na první pohled velmi rozdílné a jejich zastánci vystupují zpravidla antagonisticky, je možné oba přístupy určitým způsobem kombinovat. Rigorózní metodiky je možné odlehčit a aplikovat v jejich rámci některý z agilních přístupů. Velmi zdařilý popis aplikace základních principů agilních metodik v metodice RUP je možné nalézt v [Kroll]. Dalším příkladem propojování rigorózních a agilních metodik je aplikace metodiky *Agilní modelování* v RUP, která je popsána v [Ambler,2001AM]. Na druhé straně, pokud potřebujeme použít agilní metodiky na větší projekty či projekty větší důležitosti, je třeba je více formalizovat, zařadit více dokumentace apod. Agilní metodiky jsou v převážné většině zaměřeny na vývoj nového

řešení. V poslední době se objevuje snaha aplikovat agilní přístupy i na úpravy řešení a integraci řešení a stejně tak na některé metodiky patřící do kategorie globálních metodik.

Porovnání rigorózních a agilních metodik		
hledisko	*rigorózní metodiky*	*agilní metodiky*
náplň metodiky	procesy, zaměřují se na explicitní znalost a pohlíží na lidi jako na sekundární faktor	praktiky, zaměřují se na „tacit“ znalosti, chápou lidi jako klíčové faktory úspěchu
podrobnost metodiky	procesy a činnosti jsou popsány velmi podrobně	definována tzv**.** sotva dostatečná metodika**,** která se zaměřuje na činnosti, které vytvářejí hodnotu a eliminuje činnosti, které hodnotu nepřinášejí
kvalita	zaměření na kvalitu procesů a předpoklad, že kvalitní procesy povedou ke kvalitnímu výsledku	zaměření na hodnotu pro zákazníka a vysokou kvalitu produktu
předvídatelnost	předpokládá předvídatelnost budoucnosti, důraz na anticipaci (sběr požadavků předem, plánování předem)	předpokládá nepředvídatelnost budoucnosti, důraz na adaptaci na změny (přírůstkové shromažďování požadavků, plánování pro iteraci)
změny	změny podléhají řízení změn a je snaha změny minimalizovat	snaha změny umožnit a využít je, umožňují zákazníkům přehodnotit své požadavky s ohledem na nové znalosti
definovatelnost procesu vývoje software	vývoj software je definovaný proces, je možné jej bez problémů opakovat	vývoj software je empirický proces, nemůže být konzistentně opakován, ale vyžaduje konstantní monitorování a adaptaci
hodnota pro zákazníka	předpoklad, že dobré procesy vedou k dobrým výsledkům, je příliš zaměřen na vlastní procesy, ne na výsledky pro zákazníka.	nejvyšší prioritou je uspokojovat zákazníka
participace zákazníka na projektu	jen v počátečních a koncových fázích, po podpisu dokumentu specifikace požadavků řízení přebírá tým technologických pracovníků	přesun nositele řízení z týmu na zákazníka, zákazník je řídícím subjektem během celého projektu, při každé iteraci zákazník může měnit priority funkcí
rozsah řešení	vývojáři se snaží do systému zabudovat všechny funkce, které by mohl zákazník v budoucnu potřebovat.	pouze požadované funkce, požadavek minimalizace
vztah zákazník–vývojář	zajištěn smluvně, nedůvěra	důvěra a spolupráce
lidský faktor	sekundární, dokumentačně zaměřené procesy se snaží vykázat lidi do role zaměnitelné součástky	primární, využívá individualit a silných stránek lidí
kvalifikace lidí	stačí standardní jedinci	důraz na schopnosti, znalosti a dovednosti lidí
specialisté x generalisté	požadavek úzké specializace lidí	požadavek integrace znalostí a stálé kooperace, sdílení znalostí v týmu, týmové řešení problému, spíše generalisté než specialisté
způsob řízení	tradiční způsob řízení je formován na základě nedůvěry, direktivní řízení, kontroly	vůdcovství a spolupráce, je formováno na důvěře a respektu
význam programování při vývoji SW	důraz a hodnota jsou kladeny na architekturu, požadavky a návrh, kódování a testování jsou chápány jako činnosti s nízkou „konstrukční“ hodnotou.	důraz na programování jako činnost přinášející hodnotu
jednoduchost	spíše složité řešení, které se snaží obsáhnout i budoucí požadavky	důraz na jednoduché řešení žádné zabudovávání budoucích požadavků

Porovnání rigorózních a agilních metodik		
hledisko	*rigorózní metodiky*	*agilní metodiky*
jednoduchá x složitá pravidla	metodiky se snaží popsat vše, s čím se může vývojový tým setkat.	obsahují generativní pravidla – minimální množinu věcí, které musíte dělat ve všech situacích.
modelování	velký důraz na modelování, zejména modelování předem – big design in front of , potom se zmrazí požadavky	agilní modelování, při modelování nejde o model jako takový, ale o akt modelování, smyslem modelování je komunikace
forma komunikace	převážně písemná	důraz na komunikaci tváří v tvář
dokumentace	rozsáhlá dokumentace	podstatná není dokumentace, ale pochopení
způsob vývoje	spíše vodopádový, případně iterativní a přírůstkový s dlouhými iteracemi	přírůstkový vývoj s velmi krátkými iteracemi
ekonomika	zdroje bývají proměnnou veličinou, která zpravidla roste	snaha vždy realizovat nejvyšší hodnotu z daných peněz, cílem je hodnota pro zákazníka, ne perfektní systém, hodnota je kombinací funkcí produktu, které odpovídají potřebám zákazníka v určitý čas a za určitou cenu

tabulka 5.4: Porovnání rigorózních a agilních metodik

6 Architektura IS/ICT

V současném vysoce konkurenčním prostředí představuje IS/ICT rozhodující faktor, který ovlivňuje úspěšnost organizací. Pokud má IS/ICT plně podporovat podnikové procesy a realizovat jejich potenciál, je třeba definovat architekturu IS/ICT, která bude umožňovat integraci prvků. V této kapitole je vymezen pojem architektura IS/ICT a jsou uvedeny techniky pro její zachycení. Podrobněji jsou charakterizovány dva současné nejvýznamnější trendy - *Modelem řízená architektura* a *Architektura orientovaná na služby*.

6.1 Charakteristika architektury IS/ICT

Architektura IS/ICT představuje významný faktor úspěšného IS/ICT. Význam architektury IS/ICT bývá odvozován od role architektury ve stavebnictví a urbanistice. Chápání architektur se v mnohých publikacích liší. Metodika MMDIS rozlišuje dvě úrovně architektur IS/ICT – globální architekturu, která je hrubým návrhem celého IS/ICT, a dílčí architektury, které představují detailnější návrhy IS/ICT z hlediska různých dimenzí (funkční architektura, procesní architektura, datová architektura, technologická architektura, softwarová architektura, hardwarová architektura) [Voříšek,1997]. S rostoucím významem služeb v rámci IS/ICT bychom měli přidat i architekturu služeb.

Globální architektura (Enterprise architecture) zachycuje strukturu a procesy na úrovni celé organizace. Model globální architektury je reprezentací těchto struktur a procesů. Dobrý architektonický model zachycuje organizaci z různých pohledů a akcentuje i změny v budoucnu. Přínos vytváření globální architektury spočívá ve vyšší kvalitě výsledného řešení, větší rychlosti řešení a menších nákladech. Tím, že vytvoříme globální architekturu IS/ICT, umožníme vývojářům vytvářet systémy, které pracují navzájem konzistentně a efektivně. Systém také mnohem lépe odpovídá potřebám, protože vývojáři pracují se znalostí obecných byznys vizí a celkové infrastruktury. Výsledná řešení jsou rychlejší, protože důležitá rozhodnutí již byla realizována, byla vytvořena celková infrastruktura pro komunikaci, sdílení zdrojů apod. Globální architektura explicitně ukazuje, jakou funkcionalitu a jaká data je možné znovupoužít v rámci organizace.

Výhody vytváření globální architektury jsou zřejmé, ale přesto se v praxi setkáváme s řadou problémů. Ty pramení ze skutečnosti, že budování IS/ICT se neřídí v rámci celé organizace, ale řídí se jen jednotlivé projekty. Globální architektura často neexistuje a pokud ano, je často zastaralá nebo s ní nejsou vývojáři seznámeni.

Architektura je většinou prezentována ve formě modelů. Pro modelování architektury systému lze použít různé techniky jako například standardní modelovací jazyk *Unified Modeling Language* (UML), *Zachman Framework* a další. V současnosti se při návrhu architektury uplatňují také vzory. Architektonické vzory vyjadřují základní strukturální schéma uspořádání softwarových systémů, poskytují množinu předdefinovaných subsystémů, specifikují odpovědnosti, zahrnují pravidla pro

uspořádání vztahů mezi nimi. Vzory poskytují mocný slovník pro komunikaci mezi architekty a návrháři. Znalost a pochopení vzoru přenáší znalost a zkušenost, která je ve vzoru obsažena. [MSESP].

Pro analýzu a návrh architektury softwarového systému je možné použít také metody definované Institutem softwarového inženýrství (SEI):

- *Architecture Tradeoff Analysis Method* ^{SM} (ATAM),
- *Quality Attribute Workshop* (QAW),
- *Attribute–Driven Design* (ADD) .

Tyto metody jsou založeny na myšlence, že pro architekturu software jsou určující kvalitativní požadavky (jako výkon, bezpečnost, modifikovatelnost, spolehlivost a použitelnost). Návrh architektury spočívá v aplikaci tzv. „architektonických taktik". Architektonická taktika reprezentuje návrhové rozhodnutí, které umožňuje dosáhnout určité hodnoty atributu kvality[Bachmann,2002].

V současnosti jsou v oblasti architektur dominující dva trendy. *Modelem řízená architektura* (*Model Driven Architecture*) a *Architektura orientovaná na služby* (*Service Oriented Architecture*), které jsou podrobněji charakterizovány v následujících podkapitolách.

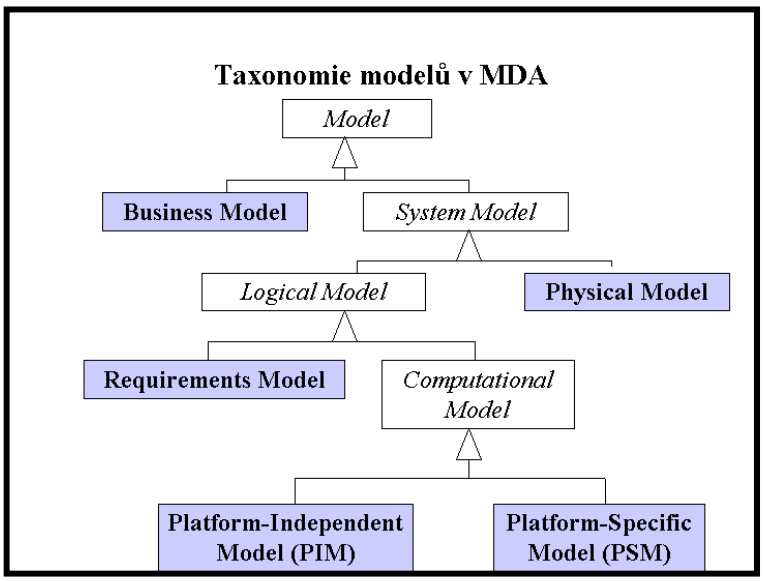
6.2 Modelem řízená architektura

Myšlenky, na kterých je založena *Modelem řízená architektura* (*Model Driven Architecture,* MDA) nejsou nijak nové. Možnost oprostit se od technologických problémů a zaměřit se na věcné problémy s sebou přinesl už vznik jazyka Cobol a rozvoj strukturovaných metod v 60. letech 20.století. Myšlenka oddělení analytického (konceptuálního) pohledu od pohledu návrhu a implementace je jedním ze stěžejních principů (*Princip tří architektur)* metodiky MMDIS.

MDA vychází ze skutečnosti, že množství změn v systému klesá s postupem na vyšší úrovně abstrakce. Dopady neustálých změn technologií je možné omezit jen na část modelu – na jeho nižší vrstvy. Při MDA vývoji se nejprve vytvoří *Platformově nezávislý model* (*Platform Independent Model* – PIM)*,* který reprezentuje věcnou funkcionalitu a chování systému. Pomocí MDA nástrojů se PIM mapuje na zvolenou platformu (například Corba, Java/EJB, XML/SOAP) a generuje se *Platformově specifický model* (*Platform Specific Model* – PSM). Nakonec se generuje implementační kód pro příslušnou technologii. MDA nástroje umožňují také zpětné inženýrství (reverse engineering), a tak je možné vytvořit modely stávajících systémů pro účely integrace aplikací. MDA generátory aplikují současně i návrhové vzory.

Na obrázku 6.1 je znázorněna hierarchie modelů používaných v rámci MDA. Modely vyznačené výplní představují konkrétní modely, které se při MDA vývoji vytvářejí. Ostatní modely jsou abstraktní a představují pouze logické členění. *Byznys model* (*Business Model*) popisuje věcné aspekty dané problémové oblasti bez ohledu na to, zda budou automatizovány, *Model systému* (*System Model*)

pak popisuje počítačový systém. *Logický model* (*Logical Model*) zachycuje logiku systému prostřednictvím modelu tříd a modelu chování, zatímco *Fyzický model* (*Physical Model*) popisuje fyzické artefakty a zdroje používané při vývoji a provozu – soubory s modely, soubory zdrojového kódu, spustitelné soubory, archivní soubory. *Model požadavků* (*Requirements Model*) popisuje počítačový systém z uživatelského pohledu a nebere v úvahu technologické aspekty řešení, zatímco *Výpočetní model* (*Computational Model*) popisuje počítačový systém včetně technologických aspektů řešení. Klíčovou úlohu v MDA plní *Platformově nezávislý model* (*Platform Independent Model*, PIM) a *Platformově specifický model* (*Platform Specific Model*, PSM). PIM představuje konceptuální model dané problémové oblasti, který je nezávislý na platformě. Platformou se rozumí určitý jazyk, technologie middleware a jim odpovídající inženýrské přístupy. Při mapování PIM na určitou platformu vznikají nejen artefakty příslušné platformy (například *Interface Definition Language,* IDL), ale také *Platformově specifický model*, který mnohem lépe zachytí sémantiku řešení pro specifickou platformu.



obrázek 6.1: Taxonomie modelů v MDA podle [Frankel]

Platformově specifický model reprezentuje jak věcnou, tak technologickou sémantiku aplikace. Je to stále UML model, ale vyjádřený v dialektu UML, který se označuje *UML profil*. *UML profil* odráží technologické prvky cílové platformy [Frankel]. MDA zahrnuje následující standardy OMG týkající se modelování:

- *XML metadata Interchange* (XMI),
- *Unified Modeling Language* (UML),
- *Common Warehouse Metamodel* (CWM) ,
- *Meta Object Facility* (MOF),

Základem MDA je modelovací jazyk UML – *Unified Modeling Language*. UML je grafická notace pro modelování, standard definovaný organizací OMG již v roce 1995. Silná stránka UML spočívá

v jeho metamodelu a v možnosti převodu modelů do XML na základě standardu XMI. UML není jeden jazyk, ale základ pro rodinu jazyků. UML obsahuje vestavěný mechanismus pro rozšíření (stereotypy a připojené hodnoty), který umožňuje definovat a používat dodatečné modelovací konstrukty a vytvářet tak dialekty UML nazývané *UML profily*. *UML profil* představuje definovanou množinu stereotypů a připojených hodnot, které rozšiřují elementy UML metamodelu a je základem pro tvorbu platformově specifických modelů. K dispozici jsou například následující *UML profily*:

- *UML profile for Corba* – defnuje mapování z PIM do PSM pro technologii CORBA,
- *UML profile for EDOC* – jazyk pro modelování spolupráce komponent a podnikových procesů nezávislý na middleware, definuje *Enterprise Collaboration Architecture* a mapování z PIM do webových služeb,
- *UML profile for EAI* – profil pro systémy komunikující pomocí zpráv.

Meta Object Facility (MOF) je jazyk pro vyjádření konstruktů modelů, tedy metajazyk. Používá stejné modelovací konstrukty pro *Diagram tříd* jako UML, a tak je možné používat pro jeho tvorbu běžné modelovací nástroje podporující UML. MOF akceptuje realitu, kdy různé modely vyjadřují různé pohledy na systém. Tyto různé modelovací jazyky je však třeba popsat jednotným způsobem a takovým univerzálním způsobem, jak popsat různé modelovací konstrukty**,** je právě MOF. Architektura MOF je tvořena 4 metaúrovněmi, jak ukazuje tabulka 6.1.

úroveň	***popis***
úroveň M3	MOF, množina konstruktů pro definici metamodelů
úroveň M2	metamodely definované pomocí konstruktů MOF, např. UML, CWM, CCM
úroveň M1	modely tvořené instancemi konstruktů M2 metamodelu, např třída Zákazník apod
úroveň M0	objekty a data, tj. instance konstruktů M1 modelu, např. zákazník Jan Novák

tabulka 6.1: Úrovně MOF

Common Warehouse Metamodel (CWM) je standard definovaný pomocí MOF. Standardizuje databázové modelovací jazyky tak, že si nástroje různých výrobců mohou vyměňovat datové modely, pravidla transformace dat, analytické specifikace.

Strategická iniciativa OMG *Model Driven Architecture* představuje renesanci myšlenek, které byly proklamovány při tvorbě software již dlouhou dobu. Převratný vývoj v technologiích a silný důraz na efektivnost prostředků vložených do informačních technologií však v současné době vytvářejí příznivé klima pro realizaci těchto myšlenek. Navíc jsou tyto myšlenky prosazovány organizací, která má dlouholeté zkušenosti s definováním standardů (UML, CORBA). Cílem MDA je zachovat investice vložené do informačních technologií tím, že se vývojáři zaměří na vytváření platformově nezávislých modelů, které reprezentují věcnou oblast a nejsou dotčeny změnami v technologiích. Z těchto modelů potom s využitím MDA nástrojů, které se dnes již začínají objevovat, budou automatizovaně generovány platformově specifické modely i vlastní implementace.

6.3 Architektura orientovaná na služby

Snaha o zefektivnění informačních technologií a důraz na přínosy v podnikových procesech vede stále více k prosazování konceptu služeb. Služby vystupují jako spojovací článek mezi podnikovými procesy a IS/ICT. Společnost Meta Group je přesvědčena, že jedním z nejdůležitějších trendů v informatice v následující dekádě bude právě orientace na služby [Metagroup,6/2003]. Otázka orientace na služby při vývoji IS/ICT není jen otázkou technologie, ale podstatně ovlivňuje i metodiky vývoje a nasazení IS/ICT. Ačkoli se technologie i ekonomické prostředí za poslední desetiletí dramaticky změnily, metodiky většinou s těmito změnami nepočítají. Většina metodik byla navržena pro prostředí, kde se aplikace vyvíjejí od začátku, ne pro prostředí založené na službách, kdy je třeba řešení poskládat z různých zdrojů.

Podle typu služby rozlišujeme aplikační služby (jejich obsahem je poskytovaná funkcionalita aplikací IS/ICT), infrastrukturní služby (realizují infrastrukturu pro služby aplikační) a ostatní služby (zajišťují koordinaci a řízení činností apod.). Současný zájem o aplikační služby je spojen zejména s technologií webových služeb. Webové služby (*Web Services*) představují evoluční vývoj, který odstraňuje nedostatky komponentových architektur (například CORBA, DCOM). Technologie webových služeb je tvořena množinou průmyslových standardů založených na XML, které specifikují komunikační protokol (SOAP), jazyk pro definici (WSDL) a registr pro publikování a vyžádání služby (UDDI) [Allen,3/2003]. Webové služby jsou tedy standardní a volně spřažené, což odděluje účastníky distribuované interakce tak, že změna na jedné straně nezpůsobí selhání na straně druhé. Kombinace těchto dvou klíčových principů znamená, že firmy mohou implementovat webové služby bez znalosti konzumentů těchto služeb a naopak. Organizace začínají nabízet určité funkčnosti ve formě webových služeb, často však pouze obalí existující komponenty. Tento přístup ke službám zdola není dostačující. Je třeba se zaměřit na přístup shora, který vyžaduje změny v architektuře IS/ICT. Je třeba zavést *Architekturu orientovanou na služby* (Service Oriented Architecture, SOA). Podstatou této architektury jsou volně spřažené (loosely coupled) na standardech založené služby [Bloomberg]. Tyto architektury kromě nových architektonických přístupů vyžadují i novou roli v organizaci – roli architekta orientovaného na služby.

Architektura orientovaná na služby je postavena na těchto klíčových principech:

- **byznys procesy řídí služby a služby řídí technologii**

  ve skutečnosti služby fungují jako abstraktní vrstva mezi podnikovými procesy a technologií
- **byznys agilita je základním požadavkem**

  schopnost odpovídat na změny požadavků je novým klíčovým požadavkem, celá architektura musí požadavek byznys agility respektovat
- **úspěšná *Architektura orientovaná na služby* se stále vyvíjí.**

Atraktivnost služeb spočívá ve zvýšení produktivity IS/ICT řešení, snížení nákladů vývoje a nasazení a zkrácení času uvedení na trh. Dalším cílem je vyšší přínos z IS/ICT řešení. *Architektura orientovaná na služby* je důležitá pro podniky, protože představuje rámec, který sjednocuje byznys model s technologiemi a realizuje funkcionalitu zajišťující efektivní podnikání. Bez *Architektury orientované na služby* se IS/ICT stávají nepropojenou kolekcí balíků, funkcí a obrazovek, které konzumují stále rostoucí zdroje na údržbu a rozvoj. *Architektura orientovaná na služby* zavádí přímý vztah mezi byznys operacemi a aplikačními službami, a tak zjednodušuje údržbu systému a jeho refaktorizaci na bázi služeb.

Seznam použité literatury

[Adhikari,1/2002] Adhikari, R.: *Testing focus boosts XP*, Application Development Trends, January 2002. Dostupný z WWW: http://www.adtmag.com/article.asp?id=5952

[AgileMet] http://www.agilealliance.org/articles/index

[Allen,1/2002] Allen, P.: *The OMG'S Model Driven Architecture*, *component development strategies*, Cutter Information Corp., January 2002. Dostupný z WWW: http://www.cutter.com/articles.html

[Allen,3/2003] Allen, P.: *Service orientation*, Expertś corner LogOn 3/2003. Dostupný z WWW: http://www.ltt.de/cgi-bin/down/download.pl?download/experts/allen-march.03.pdf [25.10.2003]

[Allen,6/2003] Allen, P.: *Service–Oriented Architecture Concepts*, Expertś corner LogOn, 6/2003. Dostupný z WWW: http://www.ltt.de/cgi-bin/down/download.pl?download/experts/allen-07.03.pdf

[Ambler,1998] Ambler, S.W.: *CRC Modeling: Bridging the Communication gap Between Developers and Users,* White Paper, 1998

[Ambler,1/2002SD] Ambler, S.W.: *Toward Executable UML*, Software Development, January 2002. Dostupný z WWW: http://www.sdmagazine.com/articles/2002/0201/

[Ambler,2001AM] Ambler, S.: *Agile Modeling and the Unified Process*, Agile Modeling, 2001. Dostupný z WWW: http://www.agilemodeling.com/essays/agileModelingRUP.htm

[Ambler,3/2003SD] Ambler, S.W.: *Something's Gotta Give*, Software Development, March 2003. Dostupný z WWW: http://www.sdmagazine.com

[Ambler,4/2002AM] Ambler, S.: *Reuse for the Real world*, Agile Modeling, April 2002. Dostupný z WWW: http://www.agilemodeling.com

[Ambler,5/2002AM] Ambler, S.: *Agile Modeling and Feature Driven Development: Not just another agile methodology*, Agile Modeling, May 2002. Dostupný z WWW: http://www.sdmagazine.com/documents/s=6977/sdmam0502/samam0502.htm?temp=IWv34-X1eX

[Ambler,6/2001] Ambler, S. W.: *The Object-Oriented Modeling Process: Process Patterns for an Architecture-Driven Approach*, An AmbySoft Inc. White Paper, 1998 Dostupný z WWW: http://www.ambysoft.com/ooModelingProcess.html

[Ambler,6/2003SD] Ambler, S.W.: *Extreme Testing*, Software Development, June 2003. Dostupný z WWW: http://www.sdmagazine.com/articles/2003/0306/

[Ambler,8/2001SD] Ambler, S. W.: *Debunking Modeling Myths*, August 2001, Software Development. Dostupný z WWW: http://www.sdmagazine.com/articles/2001/0108/0108k/0108k.htm

[Ambler,8/2002SD] Ambler, S.W.: *The Fragile Manifesto*, Software Development, August 2002. Dostupný z WWW: http://www.sdmagazine.com/articles/2002/0208/ [25.10.2003]

[Ambler,TDD] Ambler, S.: *Implicit to Formal:Validating Agile Models*, Agile Modeling, September 2002, www.agilemodeling.com

[Ambler,AMT] Ambler, S. W.: *Architecture and Architecture Modeling Techniques*, 2002. Dostupný z WWW: http://www.agiledata.org/essays/enterpriseArchitectureTechniques.html

[Ambler,AMWP] Ambler, S. W.: *Introduction to Agile Modeling*, Ronin International, Inc. White Paper, 2002. Dostupný z WWW: http://www.agiledata.org/essays/agileModeling.html

[Ambler,ASD] Ambler, S.W.: *Agile Software Development.* Dostupný z WWW: http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm

[Ambler,DP] Ambler, S.W.: *Different Projects Require Different Strategies*, 2002. Dostupný z WWW: http://www.agiledata.org/essays/differentStrategies.html

[Ambler,MDA] Ambler, S.W.: *Examining the Model Driven Architecture (MDA).* Dostupný z WWW: http://www.agilemodeling.com/shared/mda.pdf

[Bachmann,2000] Bachmann F.– Baas, L.– Klein, M.: *Quality Attribute Design Primitives* (CMU/SEI-2000-TN-017,ADA392284), SEI, Carnegie Mellon University, 2000 . Dostupný z WWW: http://www.sei.cmu.edu/publications/documents/00.reports/00tn017.html

[Bachmann,2002] Bachmann F.– Baas, L.– Klein, M.: *Illuminating the Fundamental Contributors to Software Architecture Quality*, Technical Report CMU/SEI – 2002-TR-025 http://www.sei.cmu.edu/publications/documents/02.reports/02tr025.html

[Basl,2002] Basl, J.*: Podnikové informační systémy, Podnik v informační společnosti*, Grada, Praha

2002, ISBN 80-247-0214-2.

[Basl,Majer,Šmíra] Basl, J.– Majer, P.– Šmíra, M.: *Teorie omezení v podnikové praxi, Zvyšování výkonnosti podniku nástroji TOC*, Grada, Praha 2003, ISBN 80-247-0613-X.

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

[Blankers] Blankers, T.: *Combining models and patterns: delivering on the promise of increased IT productivity*, white paper Compuware Corporation. Dostupný z WWW: http://whitepapers.informationweek.com/detail/RES/1051627431_262.html

[Bloomberg] Bloomberg, J.: *Principles of SOA*, Application Development Trends March 2003. Dostupný z WWW: http://www.adtmag.com/article.asp?id=7345

[Bruckner,2001] Bruckner, T.: *Řízení služeb informatiky v podmínkách outsourcingu*. disertační práce, VŠE, Praha, 2001.

[Bruckner,Voříšek] Bruckner, T., Voříšek, J.: *Outsourcing informačních systémů*, Ekopress, 1998, ISBN 80-86119-07-6.

[Buchalcevová, Pavličková,2002] Buchalcevová, A.– Pavlíčková J.: *Základy softwarového inženýrství - objektově orientovaný přístup*, skripta VŠE, Praha 2002, ISBN 80-2459-0268-2.

[Buchalcevová, SI2003] Buchalcevová, A.: *Model Driven Architecture jako nový přístup k vývoji i integraci aplikací*, In: Systémová integrace 2003, s.469-476, ISBN 80-245-0522-3.

[Buchalcevová, Stanovská,Šimůnek, 2002] Buchalcevová, A.– Stanovská, I.– Šimůnek, M.: *Základy softwarového inženýrství - základní témata*, skripta VŠE, Praha 2002, ISBN 80-245-0346-8.

[Buchalcevová,2002] Buchalcevová, A.: *Agilní metodiky*, In: Objekty 2002, ČZU Praha, 2002, ISBN 80-213-0947-4.

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

[CMMI] *Capability Maturity Model® Integration (CMMISM) – Version 1.1 – Staged Representation*, Technical Report CMU/SEI-2002-TR-012, The Software Engineering Institute, 2002. Dostupný z WWW: http://www.sei.cmu.edu/cmmi/

[Cockburn, MetPerProj] Cockburn, A.: *A Methodology Per Project*, Humans and Technology Technical Report, TR 99.04,1999 Dostupný z WWW: http://crystalmethodologies.org/articles/mpp/methodologyperproject.html

[Cockburn, MetSpace] Cockburn, A.: *The Methodology Space*, 1998. Dostupný z WWW: http://alistair.cockburn.us/crystal/articles/ms/methodologyspace.htm

[Cockburn,1/2002] Cockburn, A.: *Agile Software Development Joins the „Would-Be" Crowd*, Cutter IT Journal, January 2002. Dostupný z WWW: http://www.agilealliance.com/articles/articles/ACcitj0102.pdf

[Cockburn,Cryst] Cockburn, A.: *Crystal, the Manifesto, the Methodology Framework*. Dostupný z WWW: http://members.aol.com/humansandt/papers/crystal/clear/whois-c.html [

[Cockburn,Game] Cockburn, A.: *SW development as a cooperative game*, 1999. Dostupný z WWW: http://members.aol.com/humansandt/papers/asgame/asgame.html

[Cockburn,HF] Cockburn, A.: *Growth of Human Factors in Application Development*, Humans and Technology Technical Report TR 96.01, 1996. Dostupný z WWW: http://alistair.cockburn.us/crystal/articles/gohfiad/growthofhumanfactorsinsd.html

[Cockburn,JIT] Cockburn, A.: *Just-in-time methodology construction*, Humans and Technology Technical Report TR 2002.01, 2000. Dostupný z WWW: http://crystalmethodologies.org/articles/jmc/justintimemethodologyconstruction.html

[Cockburn,PARTS] Cockburn, A.: *PARTS: Precision, Accuracy, Relevance, Tolerance, Scale in Object Design*, 1997. Dostupný z WWW: http://alistair.cockburn.us/crystal/articles/parts/parts.html

[Cockburn,People] Cockburn, A.: *Characterizing People as Non-Linear, First-Order Components in Software Development*, Humans and Technology Technical Report TR 99.05, 1999. Dostupný z WWW: http://crystalmethodologies.org/articles/panlc/peopleasnonlinearcomponents.html

[Dohnal,2002] Dohnal, J.: *Řízení vztahů se zákazníky, Procesy, pracovníci, technologie*, Grada, Praha 2002, ISBN 80-247-0401-3.

[Dohnal,Pour,1997] Dohnal, J.– Pour, J.: *Architektury informačních systémů v průmyslových a obchodních podnicích*, Ekopress, 1997, ISBN 80-86119-02-5.

[Dohnal,Pour,1999] Dohnal, J. – Pour, J.: *Řízení podniku a řízení IS/IT v informační společnosti*, Praha: Vysoká škola ekonomická, 1999, ISBN 80-7079-023-7.

[Drbal,1997] Drbal, P. a spolupracovníci: *Objektově orientované metodiky a metodologie*, skripta VŠE, 1997, ISBN: 80-7079-740-1.

[Drucker,2001] Drucker, P.: *Výzvy managementu pro 21.století*, Management Press, Praha 2001, ISBN 80-7261-021-X.

[DSDM3] *Guidelenes For Introducing DSDM Into An Organisation, Evolving to a DSDM Culture*, DSDM Consortium, 1998. Dostupný z WWW: http://www.dsdm.org/kss/details.asp?fileid=33

[DSDMRUP] Tuffs, D., Stapleton, J., West D., Eason, Z.: *Inter-operability of DSDM with the Rational Unified Process*, DSDM Consortium, October 1999. Dostupný z WWW: http://www.dsdm.org/kss/details.asp?fileid=4

[FDD] *Feature-Driven Development*. Dostupný z WWW: http://www.step-10.com/Process/FDD/index.html

[Fowler,Highsmith] Fowler, M– Highsmith, J.: *The Agile Manifesto*, Software Development, August 2001. Dostupný z WWW: http://www.sdmagazine.com/documents/sdm0108a/

[Frankel,9/2003] Frankel, D.S.– et.al.: *The Zachman Framework and the OMG's Model Driven Architecture*, OMG whitepaper, September 2003. Dostupný z WWW: http://www.omg.org/mda/mda_files/09-03-WP_Mapping_MDA_to_Zachman_Framework1.pdf

[Frankel] Frankel, D. S.: *Model Driven Architecture*, *Applying MDA to Enterprise Computing*, OMG Press, 2003, ISBN 0-471-31920-1.

[Gamma,2003] Gamma, E.– Helm, R.– Johnson, R.– Vlissides, J.: *Návrh programů pomocí vzorů, Stavební kameny objektově orientovaných programů*, překlad anglického originálu, Grada, Praha 2003, ISBN 80-247-0302-5.

[Gibson] Gibson, R.: *Nový obraz budoucnosti*, 2.vydání, Management Press, 2000, ISBN 80-7261-036-8.

[Highsmith,2000] Highsmith,J.: *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, New York, Dorset House, 2000

[Highsmith, 6/2002SD] Highsmith,J.: *Does agility work?,* Software Development, June 2002. Dostupný z WWW: http://www.sdmagazine.com/articles/2002/0206/

[Highsmith,2/2000] Highsmith,J.: *Extreme programming*, Agile Project Advisory Service White Paper, February 2000. Dostupný z WWW: http://www.cutter.com/articles.html

[Highsmith,2002] Highsmith, J.: *Agile Software Development Ecosystems*, Addison-Wesley, 2002, ISBN 0-201-76043-6.

[Highsmith,7/2000] Highsmith, J.: *Retiring Lifecycle Dinosaurs*, Software Testing &Quality Engineering, July/August 2000. Dostupný z WWW: http://www.stqemagazine.com

[Humprey,PSP] Humprey, W.: *Pathways to Process Maturity: The Personal Software Process and Team Software Process.* Dostupný z WWW:
http://interactive.sei.cmu.edu/Features/1999/June/Background/ Background.jun99.htm

[Hurwitz1] *TogetherSoft Competitive Analysis*, Hurwitz Report, Hurwitz Group, 2002 http://www.tek-tips.com/gviewthread.cfm/lev2/4/lev3/29/pid/893/qid

[Charette] Charette, R.: *The decision is in: Agile versus heavy methodologies*, CUTTER Consortium. Dostupný z WWW:
http://www.cutter.com/freestuff/apmupdate.pdf

[Charette,2002] Charette, R.: *Foundations of Lean Development: The Lean Development Manager's Guide.* Vol 2, *The Foundations Series on Risk management,* Spotsylvania, ITABHI Corporation, 2002. [CD]

[Infotech] *IT Agenda 2003*, info-tech white paper. Dostupný z WWW: http://www.infotechresearchgroup.com

[Jacobson,Booch, Rumbaugh] Jacobson, I.– Booch, G.– Rumbaugh, J.: *The Unified Software Development Process*, Addison-Wesley,1999, ISBN: 0201571692.

[Jirsák,2001] Jirsák, L.: *Metodiky komponentového vývoje*, diplomová práce VŠE, 2001

[Johnson,2001] Johnson, J.–Boucher K.D– Connors K.– Robinson J.: *Collaborating on Project Success*, Software Magazine, February/March 2001

[Kazman,Klein,1999] Kazman, R.– Klein, M.: *Attribute-Based Architectural Styles* (CMU/SEI-99-TR-

022,ADA371802), SEI, Carnegie Mellon University, 1999.

[Kennedy Carter,2002b] *Configurable Code Generation in MDA using iCCG*, ref: CTN 27 v3.0, Kennedy Carter Limited, 2002. Dostupný z WWW: http://www.kc.com

[KennedyCarter 2002a] *Supporting Model Driven Architecture with eXecutable UML*, ref: CTN 80 v2.2, Kennedy Carter Limited, 2002. Dostupný z WWW: http://www.kc.com

[Kernochan,1/2003a] Kernochan, W.: *Agile Programming, Extreme Programming, and Refactoring: Not Just Another Development Fad*, Aberdeen Group, 1/2003. Dostupný z WWW: http://www.aberdeen.com

[KIT,2003] *Terminologický slovník KIT*, Vysoká škola ekonomická, Katedra informačních technologií, 2002. Dostupný z WWW: http://www.cssi.cz [12.11.2003]

[Kosek,2000] Kosek, J.: *XML pro každého*, Grada Publishing, 2000, ISBN: 80-7169-860-1

[Kroll] Kroll, P.: *The Spirit of the RUP*, the Rational edge, 2001. Dostupný z WWW: http://www.therationaledge.com/content/dec_01/f_spiritOfTheRUP_pk.html

[Kubat] Kubát, J.: *Služby HPS pro podporu Mission Critical systémů*, HP Solution News, jaro 2003.

[Lean] *Lean Programming*, Software Development Magazine, 5-6/2001. Dostupný z WWW: http://www.agilealliance.org/articles/LeanProgramming.htm

[McAndrews] McAndrews, D.R.: *The Team Software Process ^{SM} (TSP^{SM}): An Overview and Preliminary Results of Using Disciplined Practices*, Technical Report CMU/SEI-2002-0-TR-015 ESC-TR-2000-015, November 2000. Dostupný z WWW: http://www.sei.cmu.edu/publications/pubweb.html

[Metagroup,2002] *Summary of Results 2003 Worldwide IT Benchmark Report*, 2002. Dostupný z WWW: http://www.metagroup.com

[Metagroup,2003] *Best Practises in Project Estimation and Performance Management, Executive Summary*, 2003. Dostupný z WWW: http://www. metagroup.com

[Metagroup,6/2003] *IT Organizations Must Drive a Service-Based Approach Across Technical, Operational, and Business Application Domains*, 6/2003. Dostupný z WWW: http://www.metagroup.com

[Metagroup,7/2003] *Lack of project management skills is a major IT issue for many organizations*, July 2003. Dostupný z WWW: http://www. metagroup.com

[MSESP] *Enterprise Solution Patterns Using Microsoft .NET*, Microsoft Corporation, June 2003. Dostupný z WWW http://msdn.microsoft.com/practices/type/Patterns/enterprise/[15.08.2003]

[Mullins] Mullins, L.: *Management and Organisational Behaviour*, FT Prentice Hall, 2001,ISBN: 0273651471.

[Novotný,2002] Novotný, O.: *Systém evidence komponent IS/IT jako základ pro měření nákladů IS/IT*, In: Systems Integration 2002, Praha : Vysoká škola ekonomická, 2002, s. 429–436, ISBN 80-245-0300-X.

[Novotný,2003] Novotný, O.: *Aplikace metrik v referenčním modelu řízení podnikové informatiky*. disertační práce, VŠE, Praha, 2003

[OPEN] *Open, A brief overview*. Dostupný z WWW: http://www.open.org.au

[Orr] Orr, K.*: CMM versus agile development: religious wars and software development*, Cutter IT Journal, Vol.3, No.7. Dostupný z WWW:
http://www.cutter.com/freestuff/apmreport.pdf

[Paulk,2] Paulk, M.C. et al.: *Key Practices of the Capability Maturity Model*, Version 1.1, Software Engineering Institute, CMU/SEI-93-TR-25, DTIC Number ADA263432, February 1993.
Dostupný z WWW:
http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.025.html

[Paulk] Paulk, M.C. – Curtis, B. – Chrissis, M. B. – Weber, Ch. V.: *Capability Maturity Model for Software*, Version 1.1, Technical Report CMU/SEI-93-TR-024. Dostupný z WWW: http://www.sei.cmu.edu/publications/documents/93.reports/93.tr.024.html

[Polák,Merunka, Carda] Polák, J.– Merunka, V.– Carda, A.: *Umění systémového návrhu, Objektově orientovaná tvorba informačních systémů pomocí původní metody BORM*, Grada, Praha 2003, ISBN 80-247-0424-2

[Pour,2001] Pour a kol.: *Informační systémy a elektronické podnikání*, Praha: Vysoká škola ekonomická, 2001, ISBN 80-245-0227-5.

[Pree,1995] Pree, W.: *Design Patterns for Object-Oriented Software Development*, Addison-Wesley,

1995, ISBN 0-201-42294-8

[RUP] *Rational Unified Process: Best Practices for Software Development Teams*, Rational Software White Paper TP026B, Rev 11/01.Dostupný z WWW: http://www-140.ibm.com/developerworks/rational/library/253.html [12.10.2003]

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

[Scrum2] *SCRUM Software Development Process, Building The Best Possible Software*. Dostupný z WWW: http://www.controlchaos.com/scrumwp.htm

[Scrum3] *SCRUM Philosophies of Software Development*. Dostupný z WWW: http://www.controlchaos.com/philo.htm

[Schmuller] Schmuller, J.: *Myslíme v jazyku UML*, Grada, Praha 2001, ISBN 80-247-0029-8.

[Siegel] Siegel, J.: *Using OMGś Model Driven Architecture (MDA) to integrate Web services*. Dostupný z WWW: http://www.omg.org/mda/mda_files/MDA-WS-integrate-WP.pdf

[Soley,2000] Soley, P. : *Model Driven Architecture*, white paper OMG Group, 2000. Dostupný z: ftp://ftp.omg.org/pub/docs/omg/00-11-05.pdf

[SPICE] ISO/IEC TR 15504: *Information technology - Software process assessment*, International Standards Organization, 1998 Dostupný z WWW: http://www.sqi.gu.edu.au/spice/

[Stanovská, Sedláček] Stanovská, I. – Sedláček, M.: *Vývoj ASW*, interní materiál Aquasoft s.r.o, 2002

[Šťovíček,2002] Šťovíček, J.: *Analýza používání vývojových platforem a jazyků v ČR*, diplomová práce VŠE, 2002

[Szyperski,11/2002] Szyperski, C.: *The Nature Of Software - A First Cut*, LogOn Experts´Corner, Nov 02. Dostupný z WWW http://www.ltt.de/cgi-bin/down/download.pl?download/experts/szyperski.nov02.pdf

[Učeň,2001] Učeň, P. a kol.: *Metriky v informatice, Jak objektivně zjistit přínosy informačního systému*, Grada, Praha 2001, ISBN 80-247-0080-8.

[Učeň,2003] Učeň, P.: *Dimenzování informatiky jako podpůrného procesu*, In: Systémová integrace '2003, sborník mezinárodní konference, Praha, Vysoká škola ekonomická, 2003, s. 283-291, ISBN 80-245-0522-3.

[Vodáček,1999] Vodáček, L.– Vodáčková O.: *Teorie a praxe v informační společnosti*, Management Press, Praha 1999, ISBN 80-85943-94-8.

[Voříšek,MMDIS] Voříšek, J.: *MMDIS Princip multidimezionality a dimenze řešení IS*. Dostupný z WWW: http://nb.vse.cz/~vorisek/FILES/IT215_materialy_k_predmetu/05MMDIS-Princip_multidimezionality_a_dimenze_reseni_IS.zip

[Voříšek,1992] Voříšek, J.: *Design and Management Dimensions of Large Information Systems*, Sofsem 92, International Conference on Software Engineering, Žiar, 1992, 311-336.

[Voříšek,1997] Voříšek, J.: *Strategické řízení informačních systémů a systémová integrace*, Management Press, Praha 1997, ISBN 80-85943-40-9.

[Voříšek,2000] Voříšek, J.: *Nová dimenze systémové integrace – integrace podnikových procesů a znalostí*, In: Systémová integrace '2000, sborník mezinárodní konference, Praha, Vysoká škola ekonomická, 2000, s. 195-206, ISBN 80-245-0041-8.

[Voříšek,2001] Voříšek, J.: *Model "SPSR" – model řízení podnikové informatiky*, In: Sborník mezinárodní konference Systémová integrácia 2001, Demanovská Dolina, TU Žilina, 2001, str.5-18, ISBN 8-7100-880-X.

[Voříšek,Dunn,2001] Voříšek, J. – Dunn D.: *Management of Business Informatics – Opportunities, Threats, Solutions*, In: Proceedings of "Systems Integration 2001" conference, Praha: Vysoká škola ekonomická, 2001, ISBN 80-245-0169-4.

[Waskiewicz,2002] Waskiewicz, F.: Integrace podnikových aplikací s produktem Model Driven Architecture, OMG Information Day, Conference Proceedings, Prague, February 25,2002

[Williams] Williams, J.D.: *Managing IT in 2003*, Application Development Trends, February 2003. Dostupný z WWW: http://www.adtmag.com/article.asp?id=7343

[Zelený,Nožička] Zelený, J. – Nožička, J.: *Komponentní architektury COM+, CORBA, EJB, BEN*, Praha 2002, ISBN 80-7300-057-1.