ŽILINSKÁ UNIVERZITA V ŽILINE
FAKULTA RIADENIA A INFORMATIKY

MODELOM RIADENÁ ARCHITEKTÚRA A ONTOLÓGIE

Dizertačná práca

Kód 28360020163008

Študijný program:	Aplikovaná informatika
Študijný odbor:	9.2.9 Aplikovaná informatika
Pracovisko:	Katedra informačných sietí
	Fakulta riadenia a informatiky, Žilinská univerzita v Žiline
Školiteľ:	prof. Ing. Matilda Drozdová, PhD.

Žilina, 2016 **Ing. Boris Bučko**

Čestné vyhlásenie

Vyhlasujem, že som celú písomnú prácu k dizertačnej skúške vypracoval samostatne s použitím uvedenej odbornej literatúry.

Žilina, 2016

.....

vlastnoručný podpis

Poďakovanie

Ďakujem prof. Ing. Matilde Drozdovej, PhD. za cenné rady, pripomienky a odborné vedenie pri vypracovaní písomnej práce k dizertačnej skúške.

Abstrakt

BUČKO, Boris, Ing.: Modelom riadená architektúra a ontológie [Dizertačná práca]. Žilinská univerzita v Žiline, Fakulta riadenia a informatiky, Katedra informačných sietí. Školiteľ: DROZDOVÁ Matilda, prof. Ing., PhD.; Žilina: FRI ŽU, 2016, 99 s.

Cieľom tejto dizertačnej práce je navrhnúť architektonický rámec riešenia informačného systému pre riadenie znalostí v organizácii použitím princípov modelom riadenej architektúry (MDA) a ontológií ako modelovacieho jazyka v rovinách CIM (Computer Independent Model) a PIM (Platform Independent Model). Prvá kapitola analyzuje stav riešenej problematiky z pohľadu vývoja IS ale aj z pohľadu ontológie pri vývoji IS. Ďalej je v práci venovaná pozornosť metodike riešenia, po ktorej nasleduje riešenie zadaného cieľa. Toto riešenie sa zaoberá pohľadom na MDA v rámci ontologického inžinierstva, návrhom architektonického rámca pre znalostné systémy a jeho využitím. Posledná časť riešenia pojednáva o využití ontológie ako modelovacieho jazyka na úrovni CIM a PIM v rámci MDA.

Kľúčové slová: vývoj informačných systémov, modelom riadená architektúra, ontológie, znalosti

Abstract

BUČKO, Boris, Ing.: Model driven architecture and ontologies [Dissertation Thesis]. The University of Zilina, The Faculty of Management Science and Informatics, Department of Information Networks. Tutor: DROZDOVA Matilda, prof. Ing., PhD.; Zilina: FRI, University of Zilina, 2016, 99 p.

The aim of this thesis is to propose an architectural framework for information system solutions for knowledge management in the organization by using the principles of model-driven architecture (MDA) and ontology as a modeling language in the CIM (Computer Independent Model) and PIM (Platform Independent Model) levels. The first chapter analyzes the area of IS development but also using of ontology in the development of the IS. Furthermore, the work focuses on the methodology, followed by the specific solution. This solution addresses the view of MDA within the ontological engineering and proposal of architectural framework for knowledge-based systems. The last section is focused on using ontology as a modeling language in CIM and PIM levels within MDA.

Keywords: information systems development, model driven architecture, ontologies, knowledge

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

BPMN	Business Process Modeling Notation
BRMS	Business Rule Management System
CASE	Computer Aided Software Engineering
CBR	Case Base Reasoning
CIM	Computation Independent Model
DBS	Databázový systém
DFD	Data Flow Diagram
IDEF5	Integrated Definition for Ontology Description Capture Method
IEEE	Institute of Electrical and Electronics Engineers
IKT	Informačno komunikačné technológie
IM	Imlementation Model
IS	Informačný systém
IT	Informačné technológie
KMi	Knowledge Media Institute
MBE	Model Based Engineering
MDA	Model Driven Architecture
MDE	Model Driven Engineering
MDD	Model Driven Development
MnM	Mechanizmus extrakcie informácií založený na ontológii
MOF	Meta Object Facility
OCL	Object Constraint Language
ODS	Ontology Driven Systems
OES	Ontology Enhanced Systems
OMG	Object Management Group
ONIONS	ONtologic Integration Of Naive Sources
OOSAD	Object Oriented Systems Analysis and Design
OWL	Ontology Web Language
PIM	Platform Independent Model
PSM	Platform Specific Model
QVT	Query View Transformation
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RUP	Rational Unified Process
SDLC	Systems Development Life Cycle
SOA	Service Oriented Architecture
SRBD	Systém riadenia bázy dát
SSAD	Structured Systems Analysis and Design
SW	Software
TOVE	Toronto Virtual Enterprise
UML	Unified Modeling Language
UML-A	Unified Modeling Language Activity Diagram
UPON	United Process for ONtologies

VaV	Veda a Výskum
WfMC	Workflow Managemnt Coalition
XMI	XML Metadata Interchange
XML	Extensible Markup Language
XMLNS	Extensible Markup Language Namespace
XPDL	XML Process Defintion Language
ZS	Znalostné systémy
ŽU	Žilinská univerzita

Úvod

Vývoj informačných systémov (IS) zaznamenal viac vývojových etáp, ktoré sú charakterizované tak z pohľadu ich využitia ako aj prístupu k riešeniu. Vznikali rôzne typy informačných systémov pre rôzne využitie. Čo však vo vývoji IS ešte stále chýba, je jednoznačné prepojenie požiadaviek používateľov do návrhu a následne implementácie systému. Metodiky vývoja IS začínajú vývoj prevažne špecifikáciou funkcionalít bez potrebných znalostí v kontexte implementačného prostredia. Z tohto dôvodu, nie sú vždy v dostatočnej miere akceptované také požiadavky implementačného prostredia, ktoré umožnia organizačné inovácie.

Proces vývoja informačných systémov je permanentnou úlohou softvérových architektov a vývojárov, ktorí sa snažia vyvíjať inovatívne riešenia a postupy. Na druhej strane sú to samotní používatelia, ktorých požiadavky sú výzvou k hľadaniu nových prístupov vývoja IS. Preto je v ostatnom období záujem hlavne o flexibilné a na údržbu nenáročné informačné systémy, ktoré dokážu splniť požiadavky používateľov v čo najkratšom možnom čase.

1 Analýza súčasného stavu riešenej problematiky

V tejto kapitole (a následných podkapitolách) je analyzovaný vývoj informačných systémov z historického hľadiska ale aj z hľadiska taxonómie, metodík vývoja ale aj z hľadiska modelom riadeného vývoja.

Ďalej je analyzované využitie ontológie pri vývoji informačných systémov, či už sa jedná o enterprise architektúru, získavanie požiadaviek, zodpovednosti v IS, vytváranie objektovo orientovaných modelov či dokonca využitie ontológie v optimalizačných úlohách. Následne je pozornosť upriamená na ontologické inžinierstvo a znalostné systémy.

1.1 Vývoj informačných systémov

Vývoj informačných softvérových systémov (IS) má dlhú históriu. Začal v päťdesiatych rokoch minulého storočia. V priebehu viac než päťdesiatročnej histórie boli pri vývoji používané rôzne postupy, metodiky a paradigmy za rôznym účelom vývoja IS. V počiatočných rokoch bol vývoj zameraný na výkonnosť softvéru pre spracovania dát. Využitie IS bolo hlavne na automatizáciu procesov náročných na manuálne spracovanie údajov. Aplikácie boli vytvárané v strojovom kóde alebo jazyku symbolických adries. Pretože počítače boli veľmi veľké, ďalší vývoj od šesťdesiatych rokov viedol hlavne k vývoju menších a lacnejších počítačov a vývoju vlastných softvérových aplikácií používateľmi. Náročnosť vlastného vývoja aplikácií dala v sedemdesiatych rokoch podnet k systémovému vývoju zákaznícky orientovaných aplikácií, a túto skutočnosť možno považovať za začiatok vzniku softvérového priemyslu. Novou disciplínou sa stal systémový vývoj softvéru, označovaný tiež ako inžiniering, využívaný v tom čase hlavne pri vývoji databázových systémov. Osemdesiate roky sú charakteristické využívaním mikropočítačov v organizáciách a následne rozvojom softvérového priemyslu, ktorý vytváral nové aplikácie s grafickými rozhraniami používajúcimi okná a ikony. Od deväťdesiatych rokov dochádza ku systémovej integrácii, ktorá umožňuje vytvárať veľké komplexné systémy ako súbor rôznych špecifických modulov, v ktorých sa neskôr začína používať web rozhranie. Postupne sa vo vývoji IS uplatňujú bezdrôtové komponenty a aplikácie sa využívajú ako služby.

Komplexnosť informačných systémov neustále narastá. Tento trend vyplýva nielen z technologických možností, ale aj náročnosti požiadaviek, ktoré sú kladené na ich funkcionalitu. Funkčné požiadavky sú vytvárané na základe nových požiadaviek používateľov, pre ktorých je systém vyvíjaný. Je to však iba v tých prípadoch, keď požiadavky používateľov sú biznis analytikmi resp. systémovými analytikmi správne spracované a pretransformované do funkčných špecifikácií IS. Ak táto skutočnosť nie je akceptovaná, má to za následok vznik softvérových aplikácií informačného systému, ktoré dostatočne nespĺňajú potreby organizácií a ich následná modifikácia a údržba sú zložité a nákladné. Zamerať pozornosť iba na funkcionality IS, bez hlbšej znalosti štruktúry a potrieb organizácie pre ktorú je vytváraný, nie je v súčasnosti pri vývoji IS možné. Prepojenie biznis pohľadu a pohľadu vývojárov konkrétneho softvérového IS je preto stále aktuálnou témou výskumu.

1.1.1 Taxonómia IS

Množstvo rôznych označení informačných systémov vyvoláva dojem množstva tried alebo typov informačných systémov. Ich kategorizácia môže byť vytvorená na základe rôznych hľadísk. Pre účely výskumu koncepcií riešenia vývoja IS je postačujúce zaradenie do troch kategórií typov (resp. tried), v ktorých sú ďalšie označenia chápané ako podkategórie. Základné kategórie tried podľa [1] a príklady podtried sú nasledovné:

Transakčné informačné systémy (Transaction Processing System - TPS), kam patria:

- Rezervačné systémy (Reservation Information System - RIS)
- Informačná podpora kontaktu so zákazníkmi (Customer Information System - CIS)
- Elektronická výmena dokumentov (Electronic Data Interchange - EDI)
- Kancelárske informačné systémy (Office Information Systems - OIS)

Transakčné informačné systémy sú procesne orientované, zamerané na zber dát, ich overenie, spracovanie, uchovanie a prenos medzi činnosťami príslušného procesu, prípadne aj rôznymi transakčnými systémami.

Manažérske informačné systémy (Managemnet Information System - MIS), s členením na:

- Systémy riadenia vzťahov so zákazníkmi (Customer relationship management – CRM)
- Systémy riadenia dodávok (Supply Chain Management – SCM)
- Systémy riadenia zdrojov (Enterprise Resource Planning – ERP)

Manažérske informačné systémy sú dátovo orientované, zamerané na porozumenie vzťahov medzi dátami vytváraním a sprístupnením súhrnných dát, vytváraním rôznych modelov historických a budúcich trendov, podľa manažérskych alebo špeciálnych potrieb.

Systémy na podporu rozhodovania (Decision Support Systems – DSS), ktoré sú označované, často podľa implementačnej domény, ako:

- Exekutívne informačné systémy (Executive Information system - EIS)
- Expertné systémy (Expert systems - ES)
- Znalostné systémy (Knowledge Work Systems - KWS)
- Business Intelligent - BI

Systémy pre podporu rozhodovania sú dátovo orientované a v interakcii s používateľmi využívajú rozhodovaciu logiku, ktorá umožňuje identifikovať problém a hľadať, hodnotiť a posudzovať alternatívne riešenia.

Okrem uvedených tried, ktoré sú vytvorené viac z organizačného hľadiska, sú vytvárané rôzne špeciálne IS. Príkladom sú Geografické informačné systémy (Geografical Information System – GIS).

Rôzne triedy informačných systémov sú vyvíjané rôznymi prístupmi resp. paradigmami, metodikami, technikami a nástrojmi, ktorých základom je prevažne metodológia SDLC (System Development Live Cycle), aj keď počet fáz vývoja je rôzny.

1.1.2 Metodiky vývoja IS

Metodiky vývoja IS počas päťdesiatročnej histórie prešli niekoľkými etapami. Prvá z nich je SDLC, ktorá bola populárna v sedemdesiatych a hlavne v osemdesiatych rokoch minulého storočia. Vývoj, založený na jednoduchej sekvencii krokov, vychádza z princípov všeobecného systémového prístupu vývoja, ktorý má podľa [2] tri zložky: vstup, model, výstup. V SDLC sú jednotlivé fázy podľa potreby rozširované a inak označované. Napríklad v [1] je päť zložiek a sú označované Planning, Analysis, Design, Implementation, Maintenance, ale v [3] sú štyri a majú označenie Project Management and Planning, System Analysis, System Design, System Implementation and Operation. Zmena označenia jednotlivých krokov vyplýva z prispôsobenia sa použitým metodológiám.

Kým v [1] je použitá štruktúrovaná systémová analýza a návrh – SSAD, v [3] je použitá objektovo orientovaná systémová analýza a návrh - OOSAD. Od deväťdesiatych rokov minulého storočia sa vo vývoji IS začala presadzovať OOSAD a postupne nahrádza štruktúrovaný prístup v mnohých typoch systémov. Dôvod presadzovania OOSAD je zrejmý z porovnania obidvoch prístupov. Základným princípom štruktúrovanej metodológie je rozdelenie procesov a dát pri použití lineárneho prechodu medzi fázami SDLC. Rozdelenie dát a procesov pri vývoji IS je v [3] uvedené ako neprirodzené, pretože v reálnom svete dátové aspekty vecí a ich chovanie alebo procesný aspekt nie sú uvažované oddelene. Ako príklad je uvádzaný DVD prehrávač, kde atribúty reprezentujúce dáta (napr. model, cena, vzhľad) a jeho chovanie (napr. prehrávanie či podpora zobrazenia titulkov) nie sú logicky oddelené položky. Z hľadiska využitia tradičného SDLC má štruktúrovaná metodológia tiež menšiu prepojenosť medzi analýzou a návrhom IS. DFD (Data Flow Diagrams), používané v SSAD ako modely analýzy, sa dajú len analyticky/mechanicky pretransformovať do návrhových modelov a zmenu požiadaviek na systém je potrebné vykonať aj v analytických návrhoch. Tieto náročné analytické rekonštrukcie odstraňuje OOSAD tým, že definuje objekt, ktorý zapuzdruje resp. uzatvára dáta aj procesy. Takáto reprezentácia lepšie vystihuje reálny svet. Objektovo vytvárané analytické modely môžu byť transformované resp. mapované do návrhových modelov a tak sa dá zvyšovať produktivita vývoja IS. Problémom metodík OOSAD je však to, že začínajú prevažne nad množinou objektov, ktoré sú vytvárané

prevažne v UML (Unified Modeling Language) diagramoch, ale sofistikovanejšie postupy, ako nájsť objekty v riešenom IS a skontrolovať ich správnosť chýbajú.

V praxi väčšina projektov vývoja IS využíva už existujúce, všeobecné metódy na špecifikáciu procesov vývoja v rámci inžinieringu príslušnej architektúry, ktorú použijú systémoví architekti. Tieto metódy sú podľa [4] často:

- Metódami štandardov jednotlivých organizácií.
- Metódami podliehajúcimi medzinárodným či dokonca militaristickým štandardom.
- Doménovo špecifické priemyselné metódy.
- Metódy prijaté z kníh, článkov, konferencií či rôznych návodov.

Využívanie takýchto všeobecných metód pri tvorbe špecifických systémov so sebou nesie určité riziko, pretože jednotlivé IS sa v značnej miere môžu od seba odlišovať v mnohých parametroch, podľa [4] napríklad:

- **Veľkosť** - od malých podporných aplikácií až po obrovské systémy.
- **Komplexnosť** - od jednoduchých až po veľmi komplexné systémy.
- **Kritickosť resp. rizikovosť systému** - riešenie spoľahlivosti a bezpečnosti.
- **Aplikačná doména** - rôzny spôsob komunikácie, spracovanie informácií pre rôzne domény, napríklad doprava, zbraňové/military systémy,....
- **Miera znovu použiteľnosti**, ktorej produkty možno rozčleniť na:
    - COTS (Commercial off-the-shelf), produkty navrhnuté tak, aby mohli byť jednoducho nainštalované a mohli spolupracovať s existujúcimi systémovými komponentmi - operačné systémy, kancelárske balíky, programy na správu e-mailov,...
    - MOTS (Modifiable off-the-shelf, produkty typu COTS s tým rozdielom, že ich zdrojové kódy môžu byť modifikované používateľmi, dodávateľmi a tretími stranami na základe požiadaviek zákazníkov.
    - GOTS (Government off-the-shelf), produkty vyvíjané pre vládne organizácie.
- **Operačná závislosť na iných systémoch** - od samostatných stand-alone systémov až po „systémy systémov“ vzájomne prepojené na vysokej úrovni.

- **Zakomponované technológie** - vrátane rozmanitosti, nemennosti a vyspelosti týchto technológií.
- **Stupeň automatizácie** - od používateľsky ovládaných až po úplne autonómne systémy.

V poslednom desaťročí sa prístupy vývoja IS zameriavajú na väzby rozdielnych pohľadov vývoja IS, kde je významnejšie doplňované prepojenie biznis pohľadu a pohľadu vývoja softvéru. Všeobecnou paradigmou komplexného vývoja IS v organizácii, ako sociálno-ekonomickom systéme, sa stáva architektúra systému podľa normy ISO/IEC/IEEE 42010:2011, [5]. Normu tvorí popis architektúry systému, hľadiská resp. typy pohľadov architektúry, architektonické rámce a jazyky pre popis architektúry. S princípmi normy je úzko spojená metóda rámcov pre inžiniering systémovej architektúry MFESA (Method Framework for Engineering System Architectures). MFSA je podľa [4] metóda rámcov využívaná pre tvorbu projektovo špecifických/situačných metód inžinieringu systémových architektúr. Hlavným zámerom MFESA je efektívne a účinne podporiť prácu systémových architektov v inžinieringu stabilných, vysoko kvalitných architektúr vývoja IS.

MFESA pozostáva zo štyroch častí:

1. **MFESA ontológia**, ktorá jasne definuje pojmy a terminológiu inžinieringu systémovej architektúry a ich vzťahov.
2. **MFESA metamodel**, ktorý definuje základné abstraktné typy metód komponentov IS pre inžiniering architektúr systémov. Typy metód komponentov tvoria:
    a. Pracovné produkty vrátane architektúr a architektonických reprezentácií ako sú modely a dokumenty.
    b. Pracovné jednotky, vrátane aktivít, úloh a techník pre tvorbu pracovných produktov.
    c. Tvorcovia, vrátane systémových architektov, architektonických tímov a architektonických nástrojov, ktorí vytvárajú pracovné jednotky na tvorbu pracovných produktov.

3. **MFESA repozitár** obsahuje knižnicu tried znovu použiteľných metód komponentov IS. Tieto situačne - špecifické metódy komponentov IS sú určené na inžiniering architektúr systémov.
4. **MFESA metametóda** popisuje ako vytvárať na mieru metódy pre inžiniering situačne- špecifickej architektúry systému, výberom vhodných metód komponentov IS z repozitáru, a ich integráciu a väzby do novej metódy inžinieringu architektúry riešeného systému.

Uplatnením princípov normy/štandardu ISO/IEC/IEEE 42010:2011 a využitím metodiky MFESA možno odstrániť problémy objektového vývoja IS vytvorením prepojenia biznis pohľadu a softvérového pohľadu v architektúre systému. Tieto pohľady tvoria dve čiastkové architektúry, biznis architektúru a softvérovú architektúru. Každá z čiastkových architektúr má iný jazyk pre popis architektúry. Jazyky pre popis pohľadov sú v dnes najčastejšie grafické modely a modelom riadený vývoj sa tak stáva základným princípom či paradigmou vývoja IS.

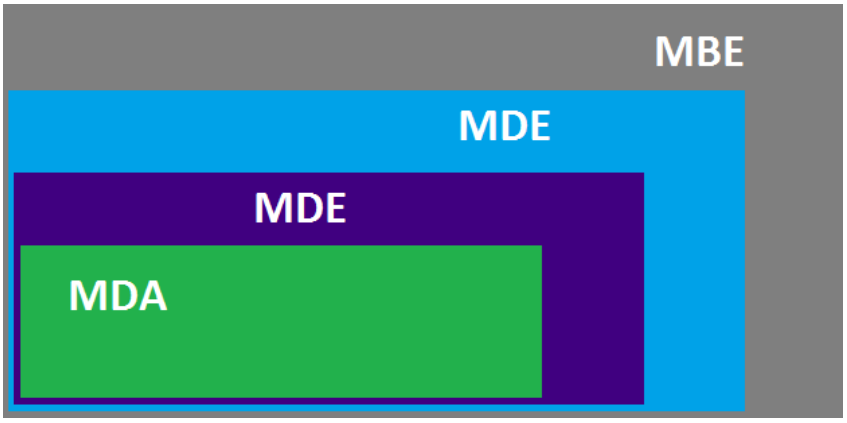
1.1.3 Modelom riadený vývoj IS

Vývojári zistili, že množstvo zmien a úprav v IS klesá úmerne s vyššou úrovňou abstrakcie vo všetkých fázach životného cyklu vývoja systému. Preto už v osemdesiatych rokoch minulého storočia bol vtedy populárny SDLC doplnený o modely v rôznych fázach vývoja a vznikla štruktúrovaná systémová analýza a návrh - SSAD, ktorá používa DFD na modelovanie procesov a ER – Entity-Relationship modely pre dátové modelovanie. Pretože štruktúrovaný prístup vývoja IS oddelene rieši tok dát medzi entitami a dátovými úložiskami systému, od polovice deväťdesiatych rokov, sa pri vývoji IS začala používať objektovo orientovaná analýza a návrh - OOSAD, reprezentovaná metodikou RUP – Rational Unified Process a novým modelovacím jazykom UML - Unified Modeling Language. Táto skutočnosť sa stala základom vytvárania ďalších paradigiem pre využívanie modelov pri vývoji IS. V priebehu vývoja vznikali rôzne termíny viažuce sa k vývoju IS s použitím modelov. Pre prehľadnosť sú nižšie uvedené rozdiely a spoločné charakteristiky nasledovných označení:

- MBE (Model Based Engineering – inžiniering založený na modeloch)
- MDE (Model Driven Engineering - modelom riadený inžiniering)
- MDD (Model Driven Development – Modelom riadený vývoj)

- MDA (Model Driven Architecture – Modelom riadená architektúra)

V [6] je uvedené, že MBE, takisto známe ako MDE či MDD, je paradigma vývoja softvéru, resp. systémov, ktorá kladie dôraz na aplikáciu princípov vizuálneho modelovania a tvorí osvedčené postupy v životnom cykle vývoja systémov (SLDC). Z toho môžeme konštatovať, že MBE je niekedy považované len za iný výraz pre MDE, lebo špecifikácie, ktoré by odlišovali MBE od MDE nie sú jednoznačne definované. V [7] je MBE popísaný ako všeobecný prístup, v ktorom modely zohrávajú dôležitú úlohu avšak nie sú „kľúčovými artefaktmi“ vývoja, pretože neriadia proces vývoja. Ako príklad je uvádzaný proces vývoja IS, kde sú vo fáze analýzy definované platformovo nezávislé modely systému, ktoré sú potom prenechané programátorom aby manuálne vytvorili kód programu, bez automatického generovania kódu. Z tohto uhľa pohľadu môžeme tvrdiť, že MBE je všeobecné a nadradené nad MDE, pretože modely neriadia proces vývoja. Ak však uvažujeme interpretácie v [7] a aj v [8] môžeme MDE chápať ako metodológiu vývoja softvéru založenú na tvorbe modelov tvoriacich abstraktnú reprezentáciu aplikačnej domény a označiť ju za podmnožinu MBE. MDD je pojem viazaný na vývoj softvérových IS alebo IT produktov. Ak však uvažujeme že MDE presahuje aktivity vývoja IS a zahŕňa kompletný proces softvérového inžinieringu založeného na modeloch, obsahujúci aj reverzný inžiniering, potom je MDD podmnožinou MDE. MDA je konkrétny prístup či paradigma MDD definovaná konzorciom Object Management Group (OMG) [9], ktorá využíva jeho štandardy modelov. Preto je MDA považovaná za podmnožinou MDD. Hierarchické porovnanie spomínaných označení, ktoré pri iných interpretáciách nemusia vždy platiť, je graficky znázornené na Obrázku 1. Je však na samotných systémových architektoch, ako si termíny vývoja IS využitím modelov interpretujú, a či uvedené pojmy striktne odlišujú. Dôležité je zmysluplné uplatnenie modelov a ich väzieb pri vývoji IS. Významné postavenie vo vývoji IS má princíp Modelom riadenej architektúry – MDA tým, že špecifikuje konkrétne pravidlá pre vývoj IS.



Obrázok 1 - Grafické porovnanie prístupov založených na modeloch

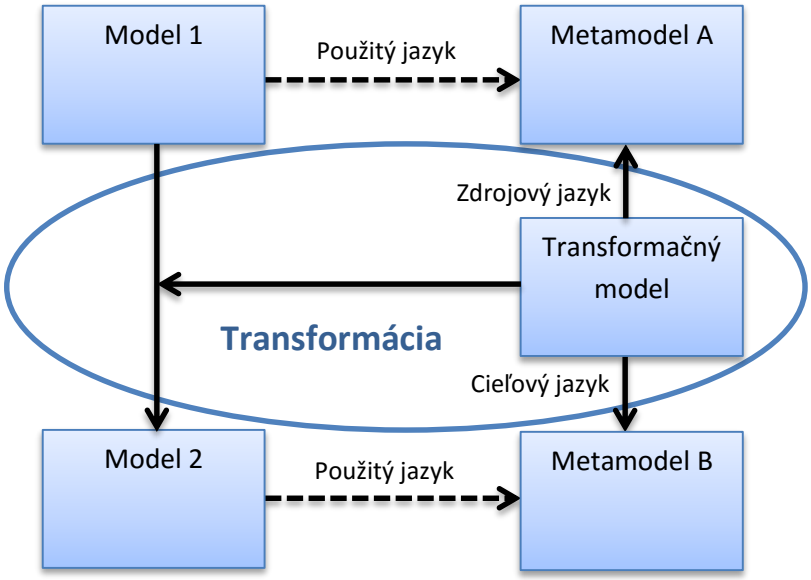
Využitie MDA pri vývoji IS

Modelom Riadená Architektúra (Model Driven Architecture – MDA) je založená na vytváraní modelov a následnej transformácií medzi nimi. Špecifikuje štyri úrovne abstrakcie:

1. Model nezávislý na počítačovom spracovaní (Computation Independent Model - CIM)
2. Platformovo nezávislý model (Platform Independent Model - PIM)
3. Platformovo závislý model (Platform Specific Model - PSM)
4. Implementačný model (Implementation Model - IM)

Tvorba modelov vyššie špecifikovaných úrovní abstrakcie je základná paradigma MDA. V prvých troch úrovniach sú vytvárané grafické modely, posledná úroveň má podobu implementovaného kódu.

Modely jednotlivých úrovní je možné postupne transformovať až do zdrojového kódu a naopak zmeny v kóde je možné inverznou MDA aplikovať spätne v modeloch. Tým sa zjednodušuje nielen vývoj, ale aj údržba takto vytvorených IS. Transformačné väzby medzi modelmi a zdrojovým kódom umožňujú udržiavať modely, kódy, súbory a dokumentáciu stále v konzistentnom stave. Transformácie medzi jednotlivými úrovňami sú dôležitým aspektom MDA. Existuje kontinuálna snaha tieto transformácie čiastočne, podľa možnosti aj úplne, automatizovať. Princíp transformačného procesu medzi dvomi úrovňami v MDA je znázornený na obrázku 2. Podľa povahy úrovní sú vytvárané transformačné pravidlá medzi zdrojovým a cieľovým modelom.



Obrázok 2 - MDA Metamodelová transformácia upravené podľa [10].

Keďže v súčasnosti, uplatnením OOSAD pri návrhu IS, sa vychádza až z modelu PIM, kladie sa najväčší dôraz na transformácie PIM – PSM v oboch smeroch. Dnes už existujú rôzne CASE nástroje a aj MDA nástroje, ktoré takúto transformáciu do veľkej miery automatizujú. Modelovaniu PIM a PSM a ich transformáciám sa venuje veľká pozornosť v mnohých prácach.

Podľa [11] môže byť transformačný proces medzi PIM a PSM realizovaný na základe:

- **Značkovania -** proces prevedenia PIM do PSM, kde sa ako prvé určí platforma. Mapovanie definuje sadu značiek (dodatočné pravidlo), ktoré sa používajú na označenie vyjadrených elementov v modeli PIM. Tieto označené elementy vytvoria „označkovaný“ PIM, ktorý je mapovaný do elementov vyjadrených v PSM.
- **Mapovania platformovo nezávislých typov na platformovo špecifické typy –** transformačné pravidlo definované v špecifikácii priamo mapuje platformovo nezávislé typy na platformovo špecifické typy. Zdrojová strana - PIM úroveň vyberá platformovo nezávislé typy, kým cieľová strana - PSM úroveň zvolí pravidlo k typu konkrétnej platformy
- **Aplikovania známych vzorov –** vzory sú používané na označenie skupiny elementov v zdrojovom modeli a ich následné mapovanie do skupín elementov

v cieľovom modeli. **Metamodelu -** Na obrázku 2 je zobrazený koncept MDA transformácie založenej na metamodeli (Metamodel A - PIM). PIM je vyjadrený v jazyku, ktorý je definovaný platformovo nezávislým metamodelom. PSM model je vyjadrený pomocou jazyka špecifického pre platformu ako Metamodel B, ktorý je určený jeho metamodelom. Transformácia definuje mapovanie medzi metamodelmi. Tento typ je vhodné použiť pre transformácie, ktoré musia premostiť zložité rozdiely medzi notáciami, v ktorých sú jednotlivé modely vyjadrené.

V praxi proces transformácie PIM - PSM môže byt oveľa zložitejší, [12]. Ide o špeciálne prípady, kedy medzi modelmi existujú také medzery, že nie je možné vykonávať priamu transformáciu. V takom prípade môžu modely pozostávať z rôznych vrstiev abstrakcie. Príkladom môže byt PIM model, ktorý sa transformuje do podrobnejšieho PIM niekoľkokrát za sebou. PIM je krok za krokom obohatený o informácie špecifické pre konkrétnu platformu, až dostaneme PSM. PSM modely môžu byt preložené do podrobnejších PSM tak, aby sa dosiahla implementačná úroveň, číže zdrojový kód. Transformácie PIM – PSM sa radia do skupiny M2M (Model to Model transformation – transformácia modelu na model), kde vstupom je model a aj výstupom je model. V MDA existujú aj iné typy transformácii: M2T (Model to Text transfromation – transformácia modelu na text), kde vstup predstavuje model a výstupom je zväčša zdrojový kód (napr. C++) alebo reverzne: T2M9. Transformácie M2T a T2M sa používajú hlavne v oblasti doménovo špecifického modelovania.

Transformácie najvyššej úrovne CIM na nižšiu úroveň PIM nie sú v OMG dokumentoch popísané. V mnohých článkoch možno nájsť viac prístupov k riešeniu [13], [14] , [15], [16]. Riešenie vytvorené na Žilinskej univerzite v Žiline, publikované v [17] umožňuje automatizovanú transformáciu PIM-CIM. PIM úroveň, reprezentovaná biznis procesmi modelovanými v notácii BPMN, je transformovaná do CIM, reprezentovanej dvomi typmi UML modelov - UML diagram prípadov použitia a model biznis entít IS reprezentovaný UML kontextovým diagramom tried. Keďže BPMN model je iba grafická podoba biznis procesu, je nutné jeho prevedenie do sémantického (textového) tvaru. Vyjadrenie sémantickej podoby modelu biznis procesov v BPMN umožňuje štandardizovaný XPDL formát. Prevedenie BPMN do XPDL formátu je vykonané

priamo biznis procesne orientovaným modelovacím nástrojom. Na prevod grafickej formy UML modelov do sémantického tvaru je použitý štandardizovaný XMI výmenný formát, ktorý vychádza z metamodelu MOF. Vytvorený prístup je založený na tvorbe automaticky vykonateľných, transformačných algoritmov, ktoré mapujú BPMN elementy, reprezentované v XPDL formáte, do elementov vybraných UML modelov reprezentovaných v MOF XMI formáte. Oba formáty, XPDL a MOF XMI sú založené na popisnom jazyku XML, preto sú transformačné algoritmy/pravidlá vytvorené prostredníctvom jazyka XSLT, ktorý je určený na tvorbu transformačných pravidiel medzi rôznymi popisnými formátmi vychádzajúcimi z XML. Automatizovaný prístup bol aplikovaný na reálnej prípadovej štúdii podpory procesu riadenia vedy a výskumu na ŽU v Žiline v projekte (Projekt ESF, 2012 "Development of human resources with the support of an integrated information system for the evaluation of scientific research results". ITMS code 26110230063). Pôvodný zámer riešenia, vytvorenie automatizovanej transformácie CIM do PIM v MDA, nebol celkom splnený. Dôvodom je skutočnosť, že vygenerované výstupné UML modely je ešte potrebné upraviť prostredníctvom ľubovoľného UML modelovacieho nástroja, pretože transformáciou dostaneme tzv. inicializačnú PIM úroveň. Preto je vhodnejšie označenie polo-automatizovaný prístup k transformácii CIM do PIM v MDA.

Pre využitie MDA pri vývoji IS je na základe (Kardos, DP) odporúčaný nasledovný postup:

- V prvej úrovni – CIM je modelovaný reálny/biznis systém danej organizácie, ktorý predstavuje implementačné prostredie vyvíjaného IS. Najčastejšie je biznis systém modelovaný procesnou mapu v určitom formálnom jazyku. V [18] je použitá pre modelovanie CIM notácia DFD, v [12] je použitá notácia BPMN. Procesná mapa je vytváraná na základe dôkladného slovného popisu procesov. Pri vytváraní inovovaných procesov je uplatňovaný princíp reinžinieringu procesov. Z procesnej mapy, ako modelu CIM roviny, sú vytvárané modely PIM.
- Rovinu PIM, použitím transformácie podľa (Kardoš DP), tvoria dva typy diagramov. use case a kontextový diagram tried, označovaný aj ako diagram biznis entít. Návrhové modely sú vytvárané v UML (Unified Modelling Language), aj keď MDA neobmedzuje formalizmus pre vyjadrenie tejto úrovne.

Iné UML modely je potrebné vytvoriť analyticky. Súbor návrhových modelov je všeobecný, nešpecifikuje konkrétnu technológiu, teda je platformovo nezávislý.

- Po špecifikácii technologických platforiem (napr. Java/PostgreSQL/C++), dostávame z návrhových modelov PIM platformovo špecifické modely PSM.
- Následne z nich, vo fáze implementácie, je získaný IM.

1.2 Ontológia pri vývoji IS

Samotný pojem „ontológia“ má svoj pôvod v oblasti filozofie, kde sa využíva v spojitosti s otázkou existencie a bytia ako celku. „*Čo je?“*, „*Čo to znamená byť*?“, „*Aký je význam bytia?*“ – to sú otázky, ktorými za zaoberá ontológia vo filozofii. Na prvý pohľad by sa mohlo zdať, že sa jedná o pomerne abstraktné pojmy a že ontológia nemá v oblasti vývoja informačných systémov svoje miesto, avšak nie je to tak. Podľa často používanej definície od amerického vedca v oblasti znalostných systémov a umelej inteligencie Grubera (Gruber, 1992) je ontológia explicitnou špecializáciou konceptualizácie. Pojem konceptualizácia všeobecne vyjadruje systém pojmov a konceptov, ktoré modelujú špecifickú časť sveta. Špecifikácia v konceptualizácii musí byť explicitne vyjadrená, formálna a tým spracovateľná výpočtovou technikou a zdieľatelná väčšiemu počtu jednotiek, ktoré spoločne uznávajú vytvorenú špecifikáciu. Iná definícia od Grubera hovorí, že ontológia je špecifikácia reprezentovaná slovníkom pre zdieľanie diskusií o doméne: definovanie tried, vzťahov, funkcií a iných objektov. Tieto definície sú pre rovinu CIM v MDA vývoja IS abstraktné a môžu sa javiť ako nie veľmi zrozumiteľné. V rovine CIM môže byť ontológia chápaná ako mapa k dátam, ktorá presne a jasne definuje ich význam – sémantiku. Ontológia dostáva tu prívlastok počítačová a jej význam je hlavne v tom, že umožňuje komunikáciu medzi informáciami a ich uložením vo forme dát. Ontológie dokážu v určitých prípadoch efektívne prepájať informácie a dáta tak, že je možné dosiahnuť lepších výsledkov ako pri použití napríklad relačných databáz, dokonca niekedy s využitím nižšieho výpočtového výkonu. Toto tvrdenie v určitých prípadoch vyzdvihuje dôležitosť softvéru pred hardvérom a možno aj čiastočne, nie však jednoznačne ponúka odpoveď na otázku *„Či je hardvér dôležitejší ako softvér“*. Vo všeobecnosti je hardvér so softvérom v takom tesnom zväzku, že nie je vhodné ani účelné vyzdvihovať dôležitosť jedného pred druhým. Avšak podľa vyjadrení [19], ktoré sa týkajú inteligentných mobilných zariadení (inteligentné telefóny, hodinky,

fit náramky,...) hardvér nie je prioritou číslo jedna. Všetko je o prístupe k dátam a o spracovaní ich obsahu. Hardvér sa časom zdokonaľuje, ale bez prístupu k dátam je jeho význam obmedzený. Tento trend sa prenáša aj ku koncovým používateľom, pretože už sa väčšinou nezaujímajú o to, aký rýchly je procesor daného zariadenia ale aká aplikácia dokáže uspokojiť ich potrebu. Softvér sa vyvíja oveľa rýchlejšie ako hardvér a softvérové platformy viac ovplyvňujú trh ako hardvérové technológie, ktorých význam je vždy spojený s príslušným softvérovým riešením. Ďalší význam využitia ontológií v aplikačnej oblasti komunikácie je medzi informáciami a znalosťami. Ontológia vo svojej podstate reprezentuje a spája informácie do znalostí presne tak ako to robí ľudská myseľ. Tento fakt by mohol výrazne posunúť vývoj umelej inteligencie, avšak teraz by bolo veľmi odvážne tvrdiť, že až do takej miery, aby dokázali stroje autonómne myslieť Aj keď IS sú iba nástroje na riešenie, využitím ontológií môžu vytvoriť vyššiu úroveň reality - umelú inteligenciu.

Ontológia ako nástroj vývoja IS je v súčasnosti využívaná v rôznych súvislostiach a fázach. Nasledujúce rešerše rôznych vedeckých článkov sú prehľadom využitia ontológie pri vývoji IS. Ich zoradenie je spracované v súlade s rovinami Modelom riadenej architektúry CIM – Computer independent model a PIM – Platform independent model.

1.2.1 Ontológia a enterprise architektúra

Tvorba IS je úzko spojená s pojmom enterprise architektúra. Enterprise architektúru možno charakterizovať ako ucelený súbor kľúčových prvkov, ktoré tvoria organizáciu. Táto definícia je však značne všeobecná. V súvislosti s normou ISO/IEC 42010 je definícia enterprise architektúry podľa [20] popísaná nasledovne:

Enterprise architektúra je prístup, koncept, prostriedok a nástroj, ktorým vyjadrujeme fundamentálne usporiadanie vzťahov medzi biznisom a jeho informačným systémom, ktoré vedie k naplneniu misie organizácie, pričom rešpektuje okolité prostredie a konzistentne dodržuje formulované princípy návrhu a rozvoja systému.

Enterprise architektúra často trpí nedostatkom sémantiky, čo sa prejavuje v problémoch komunikácie ako medzi ľuďmi tak aj medzi systémami, či dokonca priamo medzi ľuďmi a systémami. Jedným z možných riešení daného problému je ontológia podnikovej architektúry.

Ontológia enterprise architektúry pozostáva z troch vrstiev [21]:

1) Ontológia biznis výrazov.
2) Ontológia komponentov enterprise architektúry.
3) Ontológia vzťahov medzi komponentmi enterprise architektúry.

Po aplikovaní tohto prístupu sa očakáva, že ľudia a systémy budú môcť jednoznačne a presne porozumieť enterprise architektúre, čo by v konečnom dôsledku mohlo podporiť integráciu a spoluprácu v jednotlivých organizáciách ale aj medzi nimi navzájom [21].

V súvislosti s enterprise architektúrou nemožno zabúdať na modelovanie podnikových procesov a pravidiel. BPM (Business Process Management - v slovenskom preklade ako manažment biznis procesov) je možné charakterizovať ako disciplínu pre modelovanie, manažovanie, automatizáciu, monitoring a optimalizáciu procesov v podniku za účelom dosiahnutia vyššieho zisku, spokojnosti zamestnancov a celkovej prosperity podniku.

Biznis pravidlá sa používajú na rozhodovanie v rámci biznis procesov. Základným princípom je extrakcia biznis logiky z rôznorodých miest výskytu biznis pravidiel do centrálneho úložiska. Tým je umožnený nezávislý vývoj a editácia pravidiel od zvyšku informačného systému, alebo aplikácie. Vďaka takémuto prístupu môžu byť uskutočňované zmeny nad aplikačnou logikou, respektíve rozhodovacími procesmi prehľadné, efektívne a nezávislé od zvyšku aplikácie.

Podľa [22] je možné využiť ontológie aj v oblasti sémantického reinžinieringu biznis procesov pri transformovaní ontologických modelov na ne-ontologické modely biznis procesov za účelom zabezpečenia interoperability medzi skutočnými potrebami organizácie a procesnými biznis modelmi.

1.2.2 Ontologický prístup k získavaniu požiadaviek na IS

Každý vývoj IS vyžaduje vybrať nástroje, ktorými sú zisťované požiadavky používateľov v konkrétnom projekte. Analytici majú k dispozícii mnoho princípov,

nástrojov a techník. Existuje mnoho odporúčaní, ktoré sa zameriavajú na lepšie porozumenie používateľských požiadaviek. Majú rôzne označenia:

- Systémová analýza,
- Problémová analýza,
- Biznis analýza,
- Plain analýza,
- Analýza požiadaviek,
- Znalostný inžiniering.

Nie je jednoduché vybrať najvhodnejšiu techniku špecifikácie systémových požiadaviek (System Requirements Specification - SRS). Autori [23] riešia tento problém vytvorením ontológie.

Pred vytvorením ontológie je potrebný manažment požiadaviek cez dostupné techniky:

1. **Vyvolanie požiadaviek:** interview, dotazníky, pozorovanie, workshop,...
2. **Výber požiadaviek:** WinWin kooperatívne hry, prioritizácia, cenový balancing, plánový balancing,...
3. **Špecifikácia požiadaviek:** prirodzený jazyk, use case, DFD, iné štandardy,....

Následne sú vytvorené dve ontológie:

- Prvá ontológia definuje kľúčové charakteristiky súčasných techník. K výberu vhodnej techniky je potrebné poznať, aké charakteristiky sú potrebné v konkrétnej situácii.
- Druhá ontológia reprezentuje situačné charakteristiky, ktoré sú relevantné s charakteristikami techník.

Obidve ontológie sú integrované do jednej, ktorej zmyslom je zlepšenie procesu zisťovania požiadaviek.

Ďalej je potrebné zvoliť spôsob tvorby ontológie. Podľa autorov sa javí najvhodnejší Helix-Spindle model, ktorý má 3 fázy:

1. konceptuálnu fázu,
2. fázu spracovania,
3. definičnú resp. formulačnú fázu.

Autori zatiaľ spracovali konceptuálnu fázu, ktorá je rozdelená na 4 časti:

1. Popis techník, kľúčové charakteristiky, rozdiely, podobnosti. Ontológia z takýmto základom je vedená cez atribút vektor, ktorý lokalizuje každú techniku v multi-dimenzionálnom vektorovom priestore. Vektorový priestor ma 10 dimenzií.
2. Testovanie ontológie.
3. Vytváranie situačných charakteristík konceptuálnej ontológie Testovanie konceptuálnej ontológie - mapovanie situačných charakteristík do odpovedajúcich techník zberu dát..

1.2.3 Zodpovednosti v IS

Zodpovednosti plynúce z používania informačného systému sú nepochybne dôležitým aspektom avšak niekedy sa na túto skutočnosť pri vývoji IS nekladie dostatočný dôraz. Podľa [24] môže byť zodpovednosť v IS vnímaná ako odpoveď na morálne a etické aspekty IS. Na druhej strane zvyčajne nie úplne jasné čo spojenie „zodpovednosť v IS“ presne vyjadruje a akú širokú oblasť pokrýva.

Zodpovednosti v IS nesúvisia iba s prístupovými právami jednotlivých používateľov do daného IS ale aj s pátraním po vinníkoch, ktorí spôsobili problémy a v neposlednom rade aj s vyvodzovaním dôsledkov. Toto sa týka najmä informačných systémov, ktoré pracujú v prostredí, kde sú napríklad:

- Spracované citlivé dáta a súkromné informácie.

- Ukladané intelektuálne vlastníctvo.
- Výpadky napájania môžu spôsobiť problémy či dokonca život ohrozujúci stav.

Pri návrhu IS by sa systémoví architekti nemali „pozerať na používateľov cez IS“ ale naopak, mali by sa „pozerať na IS cez konkrétnych používateľov“, pre ktorých je IS vyvíjaný a tak do systému vniesť kontrolu a celkovú správu zodpovedností. Využitie ontológie na poli zodpovedností v IS takisto poskytuje priestor na ďalší rozvoj avšak opäť sa jedná o pomerne širokú problematiku, kde existuje množstvo výskumných úloh.

1.2.4 Poloautomatizované napĺňanie ontológie z textu

Samotné napĺňanie ontológie znalosťami je takisto dôležité, avšak automatizovať tento proces je neľahká úloha. Výskumníci z Knowledge Media Institute (KMi) v Británii sa týmto problémom zaoberali a vyvinuli MnM – mechanizmus extrakcie informácií založený na ontológii. Systém je schopný extrahovať pravidlá z trénovacích súborov a potom tieto pravidlá aplikovať na neznáme texty a následne naplniť ontológiu znalosťami. Ako zdroj dát poslúžil elektronický spravodajca, ktorý je vyžívaný v KMi už niekoľko rokov.

Bolo potrebné navrhnúť ontológiu, ktorá vyhovuje dianiu v KMi. Autori za týmto účelom vytvorili Udalostnú ontológiu (Event ontology) na zachytenie udalostí, ktoré sa dejú v KMi.

MnM vyžíva nástroj Amilcare na extrakciu informácií vyvinutý na univerzite v meste Sheffield. Autorom sa podarilo polo automatizovať proces napĺňania ontológie z článkov elektronického spravodajcu. Autori [25] na základe dosiahnutých výsledkov vyvodzujú nasledujúce závery:

1. Jedným z hlavných problémov bolo, že MnM vyžaduje aby používatelia pridávali anotácie k dokumentom (článkom). Ak nebudú anotácie vytvorené správne, extrakcia pravidiel takisto neprebehne správne.

2. Amilcare, systém na extrakciu informácií, je schopný rozpoznať inštancie konceptov a hodnoty, avšak nie je schopný definovať vzťahy medzi nimi. ak dokument obsahuje viac ako jednu inštanciu konceptu, Amilcare nie je schopný priradiť správne vlastnosti danej inštancií konceptu.

3. Samotní používatelia musia mať dostatočné znalosti na to, aby vedeli posúdiť, ktoré informácie z dokumentov sú vhodné na extrakciu za účelom vytvorenia pravidiel.

Poloautomatizované napĺňanie ontológie z textov interného spravodajcu by mohlo byť vhodným riešením pre udržovanie vnútropodnikových znalostí o zamestnancoch aj v situácií, keď je potrebné nájsť kvalifikovaného odborníka vo vlastných radoch.

1.2.5 Použitie ontológie pre vytváranie objektovo orientovaných modelov

Ontológia predstavuje štruktúrovaný popis vyhlásení a abstraktný spôsob vyjadrenia informácií domény v aplikácii. Koncepty v ontológii sú podobné s objektmi v softvérovom inžinierstve. Rozdielne sú modelovacie techniky. Ontológie používajú syntaktický popis na základe XML, XML schémy, RDF (Resource Description Framework) a RDF schémy (RDFS).

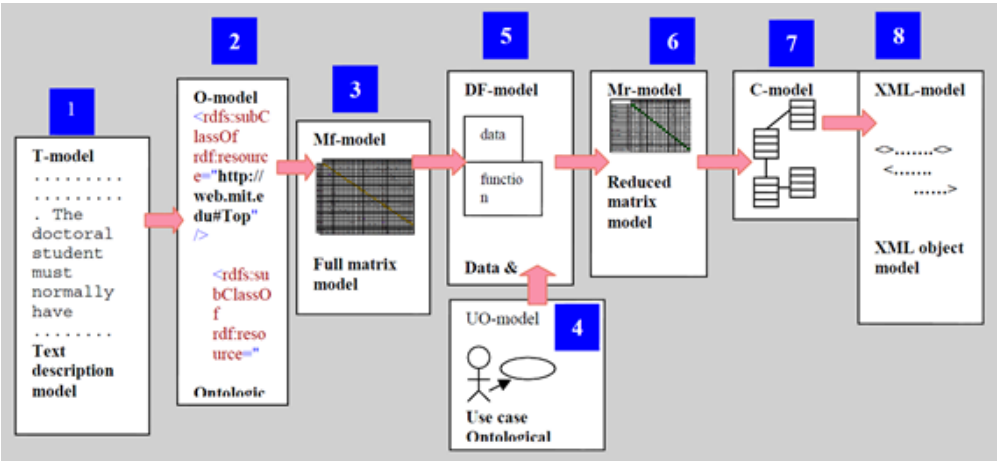
Objektové modely majú iné modelovacie techniky, pretože sú spojené koncepty premenných a abstraktných dátových typov do abstraktného typu premenných - objektu. Objekt má identitu stav a chovanie a objektové modely sú štrukturálnou reprezentáciou systému týchto objektov.

Objektovo orientovaný vývoj softvéru je akceptovaná paradigma, dostatočne podporovaná veľkým počtom metód, techník, nástrojov s výnimkou „starting point“, ktorým je identifikácia objektov a vytvorenie odpovedajúceho objektového systému modelu. Konvertovanie textového popisu systému problémovej domény do následnej/príslušnej funkčnej špecifikácie požiadaviek v objektovom modeli je spravidla odkázané na intuíciu a skúsenosti vývojárov. Všeobecne akceptované pravidlo palca je: „Ak je objekt vhodný v kontexte funkcií systému, potom patrí do systému.“ Tento problém môže riešiť ontology-driven/ontology-based prístup. Výhodou je, že objektovo orientovaný prístup má podobné paradigmy ako ontológia, čo je konceptuálna analýza domény, ktorá môže byť použitá pre rôzne aplikácie.

Abstraktné dátové typy (ADT) sú reprezentantmi objektov v objektovo orientovanom vývoji softvéru. ADT obsahujú sadu atribútov, ktoré vyjadrujú charakteristiky a formalizujú vzťahy medzi objektmi a metódami (operáciami, funkciami)

pre zavedenie do efektívneho chovania objektov, ktoré vytvárajú funkcionality systému pre praktické použitie. Základná myšlienka je implementovať ADT ako kód umožňujúci prácu objektov (inštancií tried), ktoré môžu byť dynamicky menené. Objekty sú transformované počas vývoja softvéru z reálnych vecí na koncepty a nakoniec do ADT.

Autori [26] v svojom článku popisujú konverziu textového popisu do objektových modelov. Hlavnou myšlienkou riešenia je vytvoriť vhodné transformácie, ktoré môžu byť použité ako prvky do CASE nástrojov pre objektovo orientované systémy. Prístup k riešeniu je znázornený na nasledujúcom obrázku.



Obrázok 3 - Tvorba objektovo orientovaných modelov z ontológie prevzaté z [26]

Počiatočný bod je T- model, ktorý je reprezentovaný popisom problémovej domény, z neho je vytvorený O-model (ontologický model), ktorý reprezentuje objekty. O-model je transformovaný do Full matrix (Mf) modelu, ktorý vytvára vzťahy medzi objektmi. Je to ontologický model, ktorý vyjadruje informácie o objektoch (názvy a niektoré atribúty), nie však chovanie. To je vyjadrené pomocou use case vytvorením Data Function (DF) modelu, ktorý je cez Reduce matrix (Mr) model (redukuje objekty) do Class (C)modelu (modelu tried) a z neho do XML modelu.

Uvedené riešenie bolo vytvárané za účelom identifikácie objektov systému využitím ontológie. Uvedené modely a ich vytváranie postupnými transformáciami umožní:

- transformovať požiadavky z textového popisu do objektového modelu,
- zlepšiť existujúce ontologické metódy,

- vypracovať implementačné techniky pre polo-automatické a automatické generovanie ADT,
- zlepšiť metodiky objektovo orientovaného inžinierstva.

1.2.6 Použitie ontológie pri optimalizačných úlohách

Integrácia optimalizačných a plánovacích techník pomocou formalizmu reprezentácie znalostí je výskumnou úlohou, ktorú predstavil [27]. Problém bol demonštrovaný na optimalizácií spotreby elektrickej energie v inteligentnom dome. Inteligentný dom zameraný na úsporu energie nepretržite zbiera senzorové dáta. Často krát dochádza k obmedzenej flexibilite pri plánovaní úspory energie – TV alebo mikrovlnná rúra by nemali byť odpojené od prívodu elektrickej energie, ak sú používané obyvateľmi domu. Na druhej strane zariadenia ako kúrenie alebo klimatizácia môžu byť riadené dopredu definovaným plánom. Takáto optimalizačná úloha reprezentuje typický problém úlohy o batohu, kedy je potrebné maximalizovať potenciál elektronického zariadenia s ohľadom na obmedzenia spotreby elektrickej energie. Do úlohy boli zavedené aj obmedzujúce podmienky, ako napríklad:

- Naraz môžu byť rozsvietené len 3 svetlá v dome.
- Každá chladnička a mraznička v dome musia byť vždy zapnuté.

Podstatou tohto prístupu, je zachytiť štruktúru elektrických spotrebičov spolu s ich obmedzeniami do ontológie a následne riešiť optimalizačnú úlohu. Pomerne rozšírený jazyk na zápis ontológie OWL (Web Ontology Language) neposkytuje aparát pre definovanie podmienok v rámci numerických obmedzení. OWL umožňuje definovať len obmedzenia týkajúce sa vlastností a početnosti (kardinality). Konkrétne, je možné definovať podmienku, že hodnota nejakej definovanej vlastnosti musí byť väčšia ako napríklad 5 ale nemožno definovať, že hodnota jednej vlastnosti je väčšia ako hodnota inej vlastnosti. Preto nie sú optimalizačné a plánovacie úlohy podporované jazykom OWL.

Autor vytvoril Conjuctive Query Programing jazyk, ktorý eliminuje nevýhody zápisu cez OWL.

$\mathcal{T}$ Terminology (TBox)

$Object \sqsubseteq \top$, $EnergyProfile \sqsubseteq \top$, $ElectronicDevice \sqsubseteq Object$, $CableModem \sqsubseteq ElectronicDevice$, $CentralHeatingUnit \sqsubseteq ElectronicDevice$,
$CleaningDevice \sqsubseteq ElectronicDevice$, $CookingDevice \sqsubseteq ElectronicDevice$, $Freezer \sqsubseteq ElectronicDevice$, $Fridge \sqsubseteq ElectronicDevice$,
$HiFi \sqsubseteq ElectronicDevice$, $HomeAirConditioning \sqsubseteq ElectronicDevice$, $Lamp \sqsubseteq ElectronicDevice$, $Monitor \sqsubseteq ElectronicDevice$,
$PC \sqsubseteq ElectronicDevice$, $PortableDevices \sqsubseteq ElectronicDevice$, $Desklamp \sqsubseteq Lamp$, $WashingMachine \sqsubseteq CleaningDevice$,
$Cooker \sqsubseteq CookingDevice$, $ElectricKettle \sqsubseteq CookingDevice$, $ExhaustHood \sqsubseteq CookingDevice$, $Microwave \sqsubseteq CookingDevice$,
$DVDPlayer \sqsubseteq HiFi$, $Receiver \sqsubseteq HiFi$, $Subwoofer \sqsubseteq HiFi$, $TV \sqsubseteq HiFi$,
$Room \sqsubseteq \top$, $Basement \sqsubseteq Room$, $Bedroom \sqsubseteq Room$, $Kitchen \sqsubseteq Room$, $Livingroom \sqsubseteq Room$, $MasterBedroom \sqsubseteq Bedroom$,
$\exists hasValue.\top \sqsubseteq Object$, $\top \sqsubseteq \forall hasValue.int$, $\exists inRoom.\top \sqsubseteq Object$, $\top \sqsubseteq \forall inRoom.Room$, $\top \sqsubseteq \leqslant 1\, inRoom$
$\exists hasEnergyConsumption.\top \sqsubseteq Object$, $\top \sqsubseteq \forall hasEnergyConsumption.int$, $\top \sqsubseteq \leqslant 1\, hasEnergyConsumption$,
$\exists hasCapacity.\top \sqsubseteq EnergyProfile$, $\top \sqsubseteq \forall hasCapacity.int$, $\top \sqsubseteq \leqslant 1\, hasCapacity$,
$\exists inEnergyProfile.\top \sqsubseteq Object$, $\top \sqsubseteq \forall inEnergyProfile.EnergyProfile$, $\top \sqsubseteq \leqslant 1\, inEnergyProfile$,
$\exists inCollection.\top \sqsubseteq Object$, $\top \sqsubseteq \forall inCollection.Object$, $inCollection \equiv inCollection^{-}$

$\mathcal{RB}$ Rule Base

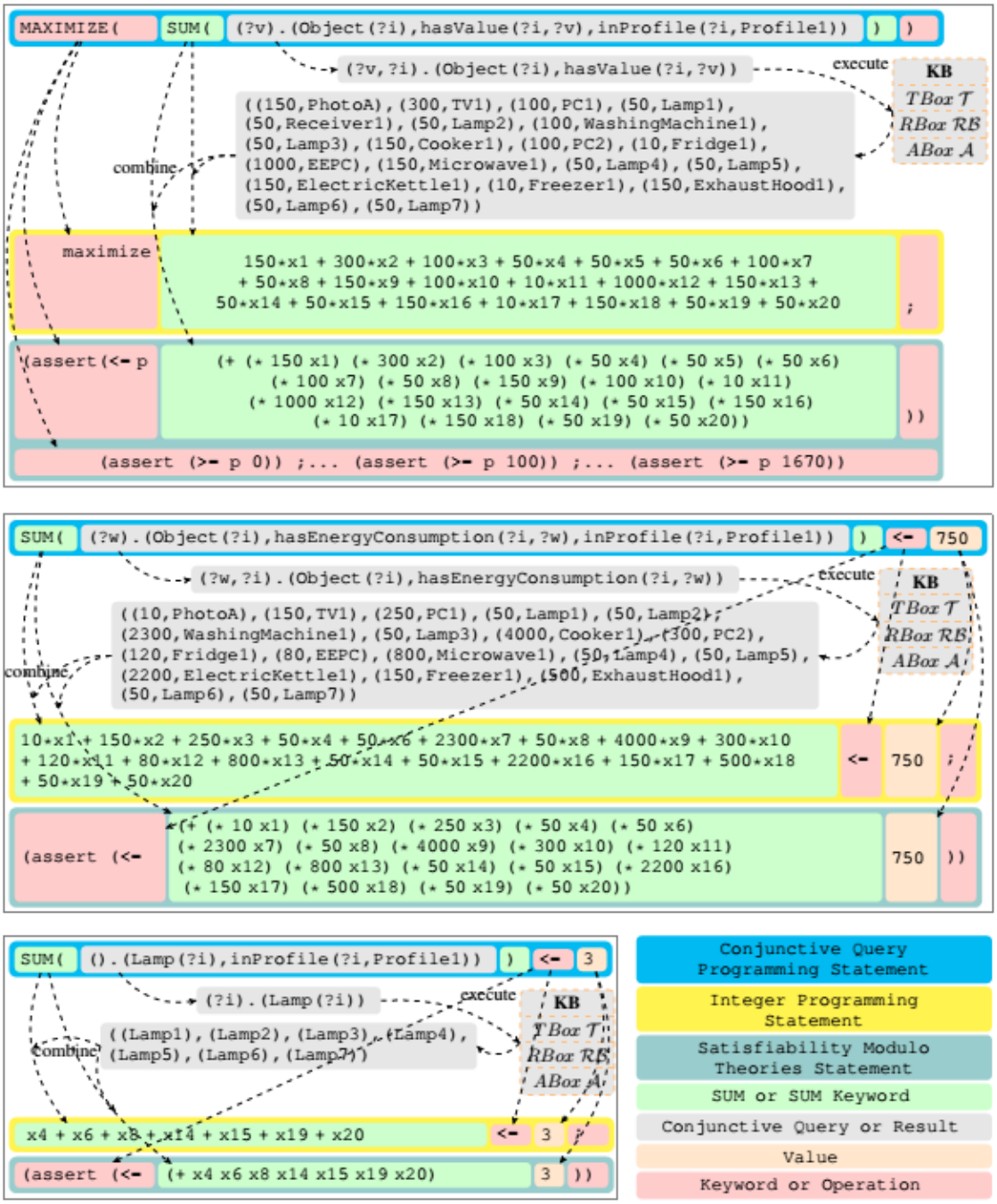
$PC(?x) \rightarrow hasValue(?x, 100)$, $CookingDevice(?x), CookingDevice(?y) \rightarrow inCollection(?x, ?y)$, $Fridge(?x) \rightarrow hasValue(?x, 10)$
$HiFi(?x) \rightarrow hasValue(?x, 50)$, $HiFi(?x), HiFi(?y) \rightarrow inCollection(?x, ?y)$, $TV(?x) \rightarrow hasEnergyConsumption(?x, 150), hasValue(?x, 250)$,
$Freezer(?x) \rightarrow hasValue(?x, 10)$, $CookingDevice(?x) \rightarrow hasValue(?x, 150)$, $Lamp(?x) \rightarrow hasEnergyConsumption(?x, 50), hasValue(?x, 50)$

$\mathcal{A}$ Instances (ABox)

$Basement(Basement1)$, $Bedroom(Bedroom2)$, $Kitchen(Kitchen1)$, $Livingroom(Livingroom1)$, $MasterBedroom(MasterBedroom1)$,
$Cooker(Cooker1)$, $Notebook(EEPC)$, $Lamp(Lamp1)$, $Lamp(Lamp2)$, $Desklamp(Lamp3)$, $Lamp(Lamp4)$, $Lamp(Lamp5)$, $Lamp(Lamp6)$,
$Desklamp(Lamp7)$, $ElectricKettle(ElectricKettle1)$, $ExhaustHood(ExhaustHood1)$, $Freezer(Freezer1)$, $Frigde(Frigde1)$, $Photo(PhotoA)$
$Microwave(Microwave1)$, $TV(TV1)$, $WashingMachine(WashingMachine1)$, $PC(PC1)$, $PC(PC2)$, $Receiver(Receiver1)$,
$inRoom(Cooker1, Kitchen1)$, $inRoom(EEPC, Livingroom1)$, $inRoom(Lamp1, Basement1)$,$inRoom(Lamp2, Bedroom2)$,
$inRoom(Lamp3, MasterBedroom1)$, $inRoom(Lamp4, MasterBedroom1)$, $inRoom(Lamp5, Livingroom1)$, $inRoom(Lamp6, Kitchen1)$,
$inRoom(Lamp7, Livingroom1)$, $inRoom(ElectricKettle1, Kitchen1)$, $inRoom(ExhaustHood1, Kitchen1)$, $inRoom(Freezer1, Basement1)$,
$inRoom(Frigde1, Kitchen1)$, $inRoom(PhotoA, Bedroom2)$, $inRoom(Microwave1, Kitchen1)$, $inRoom(TV1, Livingroom1)$,
$inRoom(WashingMachine1, Basement1)$, $inRoom(PC1, Livingroom1)$, $inRoom(PC2, Livingroom1)$, $inRoom(Receiver1, Livingroom1)$,
$hasEnergyConsumption(Cooker1, 4000)$, $hasEnergyConsumption(EEPC, 80)$, $hasEnergyConsumption(ElectricKettle1, 2200)$,
$hasEnergyConsumption(ExhaustHood1, 500)$, $hasEnergyConsumption(Freezer1, 150)$, $hasEnergyConsumption(Frigde1, 120)$,
$hasEnergyConsumption(Microwave, 800)$, $hasEnergyConsumption(PC1, 250)$, $hasEnergyConsumption(PC2, 300)$,
$hasEnergyConsumption(PhotoA, 10)$, $hasEnergyConsumption(WashingMachine1, 2300)$,
$hasValue(EEPC, 1000)$, $hasValue(PhotoA, 150)$, $hasValue(WashingMachine1, 100)$, $Profile(Profile1)$, $hasCapacity(Profile1, 750)$

Obrázok 4 - Znalostná báza manažmentu energie v dome [27]

Autorovi sa podarilo transformovať znalostnú bázu na účelovú funkciu a po vyriešení optimalizačnej úlohy výsledky opäť premietnuť do znalostnej bázy, odkiaľ je možné určiť presnú konfiguráciu elektrických spotrebičov v dome podľa zadaných podmienok.



Obrázok 5 - Použitie Conjuctive Query Programming na transformáciu znalostnej bázy na účelovú funkciu

1.2.7 Ontologické informačné systémy

Podľa [28] ontologické informačné systémy respektíve systémy využívajúce ontológiu možno rozdeliť na:

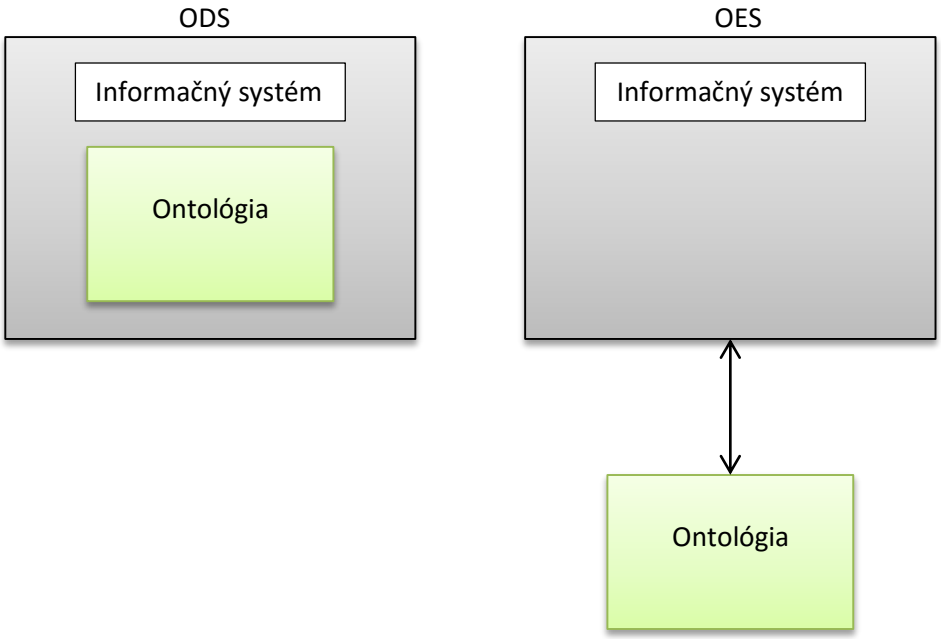
- Ontologicky riadené systémy (Ontology Driven Systems - ODS)

- Systémy rozšírené o ontológiu (Ontology Enhanced Systems - OES)

Ontologicky riadený systém je koncepciou, kedy je ontológia jedným zo základných komponentov systému a zároveň otvára nové spôsoby myslenia o ontológiách v spojení s IS, ktoré zahŕňa štrukturálne a podporné hľadisko tvorby a následného fungovania informačných systémov. Zo štrukturálne hľadiska môžu ontológie poskytnúť mechanizmus na štruktúrovanie a ukladanie obsahu IS akým sú znalosti, databázové schémy, rozhrania a aplikačné programy, ktoré môžu byť integrované v rámci IS. Z podporného hľadiska môžu ontológie výrazne pomôcť pri tvorbe nových či modifikácií existujúcich IS, lebo môžu definovať procesy, algoritmy, pravidlá a softvérové komponenty, ako súčasť vývoja IS. Ontológie a ontologicky riadené systémy sú vyvíjané a aplikované v rôznych oblastiach IT – napríklad v procesnom a systémovom modelovaní, diagnostike, podpore rozhodovania či v plánovaní.

Pri systémoch rozšírených o ontológiu, ako to už zo samotného názvu vyplýva, sa jedná o systémy, v ktorých sa ontológia používa na rozšírenie funkcionality a ontológia netvorí súčasť vývoja IS. Rozšírenie funkcionality môže mať rôzny charakter od vyhľadávania a reportov až po podporu vizualizácie.

Rozdiel medzi ODS a OES je znázornený na obr. 3.



Obrázok 6 - ODS vs. OES (zdroj autor)

Nie vždy je jednoduché či dokonca možné jednoznačne určiť, kde končia ontologicky riadené a kde začínajú ontologicky rozšírené systémy.

Napríklad podľa autorov [25], ktorí navrhli pre napĺňanie ontológie z textu mechanizmus extrakcie informácií nazvaný MnM, ide o ontologicky riadený nástroj, pretože modely a vzory na získavanie informácií z textu sú reprezentované ontológiou a ontológia je neoddeliteľnou súčasťou toho nástroja. Ak uvažujeme tento nástroj ako doplnok k existujúcemu IS, považujeme IS ako celok za ontologicky rozšírený systém.

Systémy na riadenie znalostí a zodpovedností v organizácií môžu byť ako ODS tak aj OES – záleží na zvolenej architektúre a požiadavkách na konkrétny IS (príp. na jeho časť). Či bude IS klasifikovaný ako ODS alebo OES nezáleží ani tak od implementačnej domény ale skôr od implementačného postupu. Ak IS už pri návrhu uvažuje o využití ontológie, ktorá je priamo počas vývoja implementovaná do IS, s vysokou pravdepodobnosťou bude vytvorený systém klasifikovaný ako ODS. Na druhej strane ak je ontológia doplnená do existujúceho IS, často krát sa bude jednať o OES. Toto však nie je všeobecné pravidlo.

1.2.8 Ontológie a MDA

Analýza spracovaných publikácií v kapitolách 1.2.1 až 1.2.6 vedie na nasledujúce skutočnosti využitia ontológií pri vývoji IS:

1. Znalosti zachytené v ontológií môžu byť transformované do rôznych formátov – napríklad transformovať ontológiu na účelovú funkciu pri optimalizačnej úlohe a výsledok spätne premietnuť do ontológie. (Kap. 2.7 Použitie ontológie pri optimalizačných úlohách).

2. Ontológie sú využité ako modelovacie techniky s nejednotnosťou dát, s prepájaním dát, zaoberajú sa aj modelovacími technikami pri tvorbe objektovo orientovaných modelov s využitím ontologického modelu, ktorý reprezentuje objekty problémovej domény. Tento model je následne transformovaný na model, ktorý vytvára vzťahy medzi objektmi. (Kap. 2.6 Použitie ontológie pre vytváranie objektovo orientovaných modelov.)

3. Pri spracovaní a uchovávaní znalostí (Kap. 2.5 Poloautomatizované napĺňanie ontológie z textu). Autori riešili použitím ontológie informácie o zamestnancoch tak, že extrahovali z textu interného spravodajcu také informácie, ktoré boli využiteľné napríklad pre zistenie kvalifikácie zamestnancov.

4. V modelovaní zodpovedností v IS, kde ontológia tvorí podporu reprezentácie zodpovedností tak, aby boli jasne definované a okamžite k dispozícií, bez ohľadu na to, aké iné udalosti nastanú v organizácií. (Kap. 2.4 Zodpovednosti v IS).

5. Pri získavaní požiadaviek na IS, pomocou dvoch špecifických ontológii. Prvá ontológia definuje kľúčové charakteristiky. Druhá ontológia reprezentuje situačné charakteristiky. Následne sú ontológie integrované do jednej s cieľom zlepšiť proces zisťovania požiadaviek (Kap. 2.3 Ontologický prístup k získavaniu požiadaviek na IS),

6. V procese návrhu *enterprise* architektúry, kde je ontológia rozdelená na 3 vrstvy – ontológia biznis výrazov, ontológia komponentov podnikovej architektúry, ontológia vzťahov medzi komponentmi. Cieľom je prepojiť v

enterprise architektúre systémy a ľudí, ktorí systémy využívajú. (Kap. 2.2 Ontológia a *enterprise* architektúra).

Pre modelom riadenú architektúru - MDA sú charakteristické nasledovné aspekty:

- Tvorba modelov.
- Členenie na úrovne.
- Transformácie medzi úrovňami (modelmi).
- Prechod od abstraktného ku konkrétnemu.

V analyzovaných článkoch (kapitoly 2.2 – 2.7) možno nájsť uvedené charakteristiky MDA akými je využívanie modelovacích techník, transformácie medzi modelmi či členenie na rôzne úrovne, aj keď nie vždy v zhode s vrstvami MDA, pretože ontológia môže v niektorých riešeniach vyjadrovať viac rovín MDA bez použitia transformácie. Aj keď tento fakt závisí od konkrétnej modelovanej situácie, môžeme konštatovať, že forma zápisu ontológie dovoľuje zachytiť viac rovín MDA naraz, prípade jednotlivé roviny môžu byť modelované viacerými ontológiami bez nutnosti následných transformácii, ktoré je nutné vykonať pri použití MDA.

MDA môže byť využitá aj na modelovanie ontológie, tak ako je to uvedené v [29]. Autori knihy takisto uvažujú rôzne využitia ontológii, ktoré nájdu uplatnenie nielen pri tvorbe IS:

Spolupráca – rôzni ľudia majú rôzne pohľady na rovnakú problémovú oblasť, keď spolupracujú na tímovom projekte. Toto platí najmä vtedy, keď sa stretnú špecialisti z rôznych oblastí výskumu, vývoja a technológii. Pre takýchto špecialistov poskytuje ontológia zjednotenú formu znalostí, ktorá môže byť využívaná ako všeobecná zdieľaná referencia poskytujúca priestor pre ďalší rozvoj a spoluprácu.

Integrácia – ontológie umožňujú integráciu informácií z rôznych a navzájom oddelených zdrojov. Konečných používateľov zväčša nezaujíma ako sa dostanú k informáciám. Chcú len získať informácie, ktoré potrebujú a chcú ich získať všetky. Aplikácie musia často pristupovať k rôznym zdrojom informácií, ktoré sú v rôznych formátoch a obsahujú rôznu úroveň detailov. Ak by tieto zdroje pracovali s rovnakou ontológiou, integrácia informácií by bola omnoho jednoduchšia a prirodzenejšia.

Vzdelávanie – ontológie sú takisto dobrým publikačným médiom, pretože pomocou nich je možné poskytovať prehľadné a relevantné informácie tým, ktorí sa chcú dozvedieť niečo a špecifickej doménovej oblasti.

Modelovanie – pri modelovaní inteligentných, znalostných aplikácií reprezentujú ontológie dôležité, znovu použiteľné stavebné bloky, ktoré by mohli byť zahrnuté v mnohých IS ako znalostné moduly.

Princíp modelom riadenej architektúry dokáže poskytnúť výraznú podporu a systémový prístup pri tvorbe informačných systémov. Niektoré aspekty vývoja IS, ktoré sa týkajú najmä riadenia znalostí v rovine CIM, nie je možné zabezpečiť iba pomocou MDA. Sú to hlavne tieto aspekty:

- Reprezentácia znalostí a s tým spojená tvorba znalostných IS.
- Vyvodzovanie na základe znalostí.
- Sémantické prepájanie dát.
- Efektívne modelovanie závislostí a nadväznosti dát.
- Transformácia dát a informácií na znalosti.

V týchto prípadoch je vhodné využiť ontológie súčasne s využitím princípov MDA.

1.3 Počítačové ontológie

Autori [30] použili výraz počítačové alebo výpočtové ontológie *(computational ontologies)*, ktorý reprezentuje všetky typy ontológií vyvíjané v konkrétnej doméne/disciplíne.

Dôvodom pre použitie ontológii je štruktúrovanie a zjednotenie znalostí o konceptoch, vzťahoch, axiómoch a obmedzeniach patriacich doméne tak, aby mohli byť použité v počítačovom prostredí s cieľom zvýšiť efektívnosť práce ľudí a strojov pri plnení úloh v rámci zvolenej domény. Z tohto dôvodu už dávno nie sú ontológie iba disciplínou filozofie. Jedná sa o základný predmet bádania a vývoja v oblasti informačných systémov. K tomuto záveru dospeli autori [31], pretože informačné systémy v sebe zohľadňujú znalostné artefakty, ktoré zachytávajú a reprezentujú znalosti o entitách, vzťahoch, obmedzeniach a procesoch v danej aplikačnej oblasti. Takisto

profesionáli a výskumníci z oblasti tvorby IS sa neustále stretávajú s potrebou identifikácie a reprezentovania spomínaných doménových znalostí v rámci informačných systémov.

Myšlienka zdieľania znalostí medzi ľuďmi a počítačmi prichádza s novou požiadavkou – aby ľudia aj systémy zdieľali akýsi spoločný slovník, ktorý by dokázal popísať svet. Dá sa povedať, že ontológia je mapou pre dáta, ktorá zjednocuje ich význam s ich obsahom a je vhodným spôsobom pre reprezentáciu znalostí. Ontológia si postupne získava svoje miesto v rozsiahlych moderných systémoch, ktoré využívajú znalostné bázy alebo pracujú v súlade s princípmi umelej inteligencie. To však nie je konečné využitie pre ontológie.

Ontológie sú využívané na formálnu reprezentáciu znalostí ako množina konceptov a ich vzťahov. Používajú sa na popis doménovej oblasti a jej entít. Pojmy *doména, koncept* a *sémantický vzťah* sú vysvetlené nižšie.

- **Doména** – množina, ktorá obsahuje všetky koncepty a vzťahy v danom ontologickom modeli.
- **Koncept** – definuje podmnožinu domény.
- **Sémantické vzťahy** – definujú vzťahy viažuce sa k doméne.

Znalosti v ontológií sú reprezentované pomocou trojíc – *subjekt, predikát, objekt,* kde subjekt a objekt sú koncepty a predikát vyjadruje ich sémantický vzťah. Napríklad:

spisovateľ – píše – knihu

subjekt – predikát – objekt

1.4 Ontologické inžinierstvo

Podľa [30] je ontologické inžinierstvo zamerané na hľadanie správnych odpovedí na nasledujúce kľúčové otázky:

- Aký je účel, pre ktorý sa ontológia vytvára?
- Aké zručnosti sú potrebné pri konceptualizácii a budovaní ontológie?
- Čo reprezentuje navrhnutá ontológia?

- Aká metodika je použitá pri vývoji ontológie.

V mnohých ohľadoch môže byť ontologický inžiniering prirovnaný k procesu tradičného vývoja informačných systémov, ktorý začína analýzou, pokračuje tvorbou systémových modelov, a vedie k tvorbe logického a fyzického návrhu systému. Ontologický inžiniering je v tomto smere veľmi podobný až na fakt, že analyzované domény môžu byť veľmi rozsiahle a zachytenie znalostí a mechanizmus ich reprezentácie môže byť značne odlišný.

1.4.1 Metodiky tvorby ontológii

Vytvorenie ontológie nie je triviálny proces, a preto je potrebné mať priamo postup, ktorý bol navrhnutý práve za účelom vytvorenia ontológie. Ako vhodne poznamenal [32], za posledných 20 rokov bolo vytvorených množstvo postupov pre vytváranie ontológii. Tieto metodické aktivity v posledných rokoch žiaľ výrazne spomalili svoje napredovanie. To však ale neznamená, že by bola všeobecne prijatá niektorá z metodík tvorby ontológie, ktoré boli v uplynulých rokoch vytvorené.

Niektoré z vedúcich metodík podľa [32] sú nižšie stručne uvedené od najstarších po najnovšie:

- TOVE (Toronto Virtual Enterprise) – prístup pomocou logiky prvého rádu na reprezentovanie aktivít, stavov, zdrojov, časov a nákladov v architektúre pre enterprise integráciu.
- IDEF5 (Integrated Definition for Ontology Description Capture Method) – je časť širšieho súboru metodík, ktoré boli vytvorené spoločnosťou Knowledge Based Systems, Inc.
- ONIONS (ONtologic Integration Of Naive Sources) – súbor metód zameraných na integrovanie viacerých informačných zdrojov s osobitým dôrazom na doménové ontológie.
- METHONTOLOGY – jedna zo známejších metodík pre tvorbu ontológie, napriek tomu nemá veľa známych využití.
- UPON (United Process for ONtologies) – inkrementálno-iteratívny prístup založený na UML prípadoch použitia.

1.4.2 Zostavenie ontológie

Niekoľko výskumníkov (napríklad: Borst a kol., 1997, Fernández a kol., 1997, Holsapple and Joshi, 2002, Kishore a kol., 2004b, Noy and McGuinnss, 2002, Uschold, 1996, Uschold and Gruninger, 1996, Uschold a kol., 1998) navrhlo metodiky zostavenia ontológie založené na ich osobných skúsenostiach. Hoci sú tieto prístupy ovplyvnené ich osobitým štýlom a preferenciami, autori [30] našli štúdiom týchto techník všeobecné šablóny a smery, pomocou ktorých navrhli stratégiu pre tvorbu ontológie s názvom CUE-N-ANCHOR (vo voľnom preklade ako Ukotvenie stopy). Táto stratégia sa skladá z niekoľkých bodov:

- Definovať oblasť a rozsah ontológie najlepšie ako sa len dá.
- Vykonať základnú analýzu.
- Ukotviť ontológiu a využiť stopy, ktoré budú sprevádzať vývoj ontológie.
- Vytvoriť slovník pojmov.
- Štruktúrovať predvolený slovník do osobitej ontológie s použitím crisscross stratégie.
- S využitím Cue-n-Anchor prístupu rozhodnúť o integrovaní existujúcich ontológii s aktuálne vytváranou a zhodnotiť mechanizmus formálnej reprezentácie.
- Vytvoriť formálnu reprezentáciu ontológie.

1.5 Znalostné systémy

Znalostné systémy (ďalej len ZS) sú charakteristické uchovávaním znalostí. Tento fakt vytvára zo znalostných systémov veľmi širokú doménovú oblasť. Všeobecne môžu byť znalosti uchovávané v akejkoľvek forme v súvislosti s akýmkoľvek (nielen informačným) systémom. Tak môžu byť znalosťami napríklad aj poznámky v zápisníku.

Z pohľadu vývoja znalostných softvérových systémov možno tvrdiť, že ich princípy sú veľmi podobné so všeobecnými princípmi vývoja IS. Líšia sa po väčšine len v tom, že znalostné systémy musia riešiť nielen spracovanie informácie ale aj *manažment*

znalostí – reprezentovanie, ukladanie a získavanie znalostí. Vytvorenie kategórie znalostných systémov je prirodzený vývoj a mnohé existujúce IS, by sa tiež dali považovať priamo za znalostné IS, napriek tomu, že neboli vytvárané s ohľadom na manažment znalostí. Ako príklady je možné uviezť IS verejnej správy, informačné systémy umožňujúce predaj produktov (či už sa jedná o reťazce obchodov v kamenných predajniach alebo online predaj). Svoje miesto tu má napríklad aj ekonomický software. Všetky uvedené príklady IS v počiatkoch ich tvorby neboli vytvárané za účelom manažmentu znalostí. Ako čas plynul, pribúdala nová funkcionalita na základe požiadaviek trhu a jednou z nich bola (a v súčasnosti stále je) požiadavka manažmentu znalostí.

1.5.1 Dáta, informácie, znalosti

V kontexte manažmentu znalostí je potrebné chápať pojmy znalosti, dáta a informácie. Podľa [33] dáta predstavujú kolekciu faktov, meraní a štatistík. Informácie predstavujú organizované a spracované dáta v istom časovom období (rámci) a s istou presnosťou, napríklad s odkazom na pôvodné dáta.

Znalosť resp. znalosti je možné definovať ako skutočnosť alebo stav, kedy je známe niečo na základe nadobudnutých informácií, skúseností alebo asociácií. V literatúre existuje množstvo definícií znalostí. Znalosti možno charakterizovať aj ako myšlienky, postoje a celkové chápanie, ktoré využíva jednotlivec, skupina či organizácia na svoje konanie za účelom dosiahnutia stanovených cieľov. Pri pokuse definovať znalosti netreba zabúdať na to, že ľudská myseľ je schopná vnímať znalosti racionálne ale aj intuitívne. Intuitívne znalosti boli zväčša znevýhodňované voči racionálnym, vedeckým znalostiam. V dôsledku neustáleho rozvoja vedy vznikali názory, že intuitívne znalosti vlastne nie sú znalosťami. [34]

Možno tvrdiť, že znalosti majú stále hodnotu a sú znovu použiteľné aj po uplynutí istého množstva času a takisto sú relevantné aj s historického hľadiska, zatiaľ čo hodnota informácií sa s plynutím času a bez zachovania príslušného kontextu výrazne znižuje. S plynutím času sa informácie hromadia ale znalosti sa vyvíjajú. [33]. Podľa [35] a [36] je možné znalosti rozdeliť na niekoľko typov:

- **Deskriptívne** *(Descriptive)* – obsahujú informácie o minulosti, prítomnosti, budúcnosti a zaoberajú sa znalosťami typu „*vedieť čo*“.
- **Procedurálne** *(Procedural)* – zaoberajú sa znalosťami typu „*vedieť ako*“ a špecifikujú postupnosti krokov (procedúr) ako dosiahnuť cieľ.
- **Uvažovacie/Zdôvodňovacie** *(Reasoning)* - zaoberajú sa znalosťami typu „*vedieť prečo*“ a zhodnocujú závery, ktoré sú platné pre daný súbor okolností.
- **Prezentačné** *(Presentation)* – uľahčujú komunikáciu a vzťahujú sa na spôsob prenosu znalostí.
- **Lingvistické** *(Linguistic)* – interpretujú komunikáciu, po tom ako táto komunikácia prebehla.
- **Asimilačné** *(Assimilative)* – napomáhajú spravovať bázu znalostí samotným zlepšovaním existujúcich znalostí.

Prvé tri typy pokrývajú základné znalosti, ktorými disponuje organizácia na úrovni vykonávania vlastných biznis procesov. Ďalšie tri typy zabezpečujú komunikáciu, chápanie a učenie sa znalostí za účelom ich použitia.

Samotné znalosti tvoria elementárnu súčasť každej organizácie no sú často krát podceňované. Nie je kladený dostatočný dôraz na systémové uchovávanie znalostí v takej forme, aby mohli byť jednoducho dostupné a spravované naprieč celou organizáciou. Vo všeobecnosti je možné tvrdiť, že fungovanie každej organizácie je vo výraznej miere postavené na znalostiach. Z tohto dôvodu je potrebné jednotlivé znalosti spravovať.

Organizácie vynakladajú nemalý obnos finančných zdrojov aby si vytvorili rámec, ktorý by zastrešoval manažovanie dát, informácií a znalostí.

V súčasnosti, každá organizácia produkuje oveľa viac obsahu ako to bolo v minulosti. Dôsledkom toho je aj fakt, že spoločnosť sa stretáva s informačným presýtením. Množstvo štruktúrovaných a neštruktúrovaných informácií narastá exponenciálne a vzniká potreba hľadať spôsoby ako tieto informácie spájať, selektovať a agregovať tak, aby poskytli relevantný výstup v čo najmenšom možnom čase. Neustále dochádza k firemným fúziám a plynulá integrácia pôvodného obsahu spájaných

organizácií cez informácie, dáta a znalosti do novej štruktúry je takisto veľmi dôležitá. [37]

ZS pracujú s explicitnou reprezentáciou znalostí a využívajú rozdielne druhy znalostí o určitej aplikačnej doméne. Vývoj ZS mnohokrát komplikujú problémy s prevzatím či tvorbou znalostí. Tento problém sa zvykne nazývať aj „zúžené hrdlo prevzatia znalostí“ a vyznačuje sa (Mathias, 2008):

- Úzkymi zónami: Prenosové kanály určené na prenos znalostí organizácie zo zdroja (doménoví experti, dokumenty, procesy) sú pomerne úzke.
- Oneskoreným získaním znalostí: Pomalá rýchlosť získania znalostí je často sprevádzaná oneskorením medzi okamihom, kedy sú znalosti vytvorené a kedy môžu byť využívané.
- Nepresnosťou znalostí: Doménoví experti robia chyby a takisto aj datamining môže vytvoriť chybné predpoklady. Údržba môže zaviezť nepresnosti alebo nezrovnalosti do pôvodne správnych znalostných báz.
- Údržbovou pascou: Ako znalosti v znalostnej báze narastajú, rastú aj požiadavky na údržbu. Takisto predchádzajúce údržby a aktualizácie, ktoré boli vykonané „horúcou ihlou“ a s nedostatočnou starostlivosťou robia budúcu údržbu ešte náročnejšou.

Niektoré prístupy sa snažia spomínaný problém eliminovať tvorbou konceptov a reprezentáciou znovu použiteľných znalostných komponentov. ZS je vnímaný ako systém, ktorý pozostáva zo samostatných, no napriek tomu z prepojených, navzájom spolupracujúcich komponentov. Medzi tieto komponenty môžu patriť (Díaz-Agudo & Gonzáles-Calero, 2007):

- Doménové znalosti.
- Štruktúrované úlohy a postupnosti krokov.
- Metódy riešenia problémov (Problem Solving Methods), ktoré reprezentujú všeobecne vyskytujúce sa doménovo nezávislé stratégie riešenia problémov.

S pojmom manažment znalostí a systémami na riadenie znalostí sú veľmi úzko spojené ontológie. Použitie ontológie pri vývoji systémov sa označuje pojmom ontologické systémy. Ontológia poskytuje aparát, ktorý je schopný pracovať so samotnými znalosťami na pomerne nízkej úrovni a tým podporuje prácu s informáciami a dátami na vyšších úrovniach IS.

1.5.2 Manažment znalostí

Je dôležité aby organizácie a spoločnosti mali prehľad o svojich znalostiach a aby „vedeli, čo vedia“. Manažment znalostí v praxi znamená riadenie korporátnych znalostí a duševného majetku, čo môže zlepšiť celý rad výkonnostných indikátorov organizácie a vytvoriť pridanú hodnotu tým, že organizácia môže konať s využitím dostupných znalostí efektívnejšie a rozumnejšie ako predtým. (Gupta, Iyer, & Aronson, 2000). Znalostný manažment zohráva významnú úlohu na strane spracovania a poskytovania znalostí subjektom, ktoré tieto znalosti vyžadujú, či už sa jedná o zamestnancov alebo o samotné informačné systémy.

Podľa [37] je hlavným účelom ontológie pri vývoji IS na riadenie znalostí – znalostných systémov, umožniť komunikáciu medzi počítačovými systémami spôsobom, ktorý je nezávislý na jednotlivých systémových technológiách, architektúrach IS a aplikačných doménach. Kľúčovou zložkou, ktorá tvorí ontológiu sú slovníky elementárnych pojmov a s nimi spojená dôkladná špecifikácia toho, čo tieto pojmy znamenajú. Súbor vzťahov medzi pojmami naviguje pracovníkov využívajúcich znalostné systémy tak, aby sa mohli jednoducho orientovať v definovanom sémantickom priestore. Jednotlivé kategórie a úrovne poskytujú k dopytovaným informáciám významovo platný obsah. V rámci kategórie sa môže vyhľadávanie v ontológií presúvať od jedného vyhovujúceho konceptu k inému, učiť sa o podobných významoch, alebo začať úplne nové vyhľadávanie v širšej oblasti výsledkov a postupne prechádzať k čoraz viac špecifickým inštanciám jednotlivých konceptov. S využitím tohto prístupu sa môžu ontológie využívať v rámci IS aj na:

- Kategorizáciu dokumentov.
- Indexovanie dokumentov.
- Prehľadávanie dokumentov.

- Spracovanie dopytov používateľov.
- Verifikáciu výsledkov.

1.5.3 Znalostné inžinierstvo

Podľa [38] znalostné inžinierstvo (disciplína aj povolanie) vyplynulo z praktickej aplikácie dlhodobého výskumu v oblasti umelej inteligencie a inteligentných systémov. Pojem znalostné inžinierstvo sa odkazuje, na vývoj systémov, ktoré využívajú znalosti na riešenie najrôznejších výpočtových problémov pomocou transformácie dát do znalostí/vedomostí.

Slovník TechnoPedia definuje pojem *znalostný inžinier* ako profesionála zaoberajúceho sa vedou budovania vyspelej logiky do počítačových systémov, aby sa pokúsil simulovať rozhodovanie človeka a kognitívne úlohy na vysokej úrovni. V znalostnom inžinierstve sa jedná o transfer ľudskej logiky a znalostí do technológií.

Tvorba znalostných BRMS (Business Rule Management System) či CBR (Case Base Reasoning) systémov v istom zmysle takisto podlieha znalostnému inžinierstvu.

BRMS– systémy na správu biznis pravidiel. Jedná sa o prístup založený na pravidlách resp. na biznis pravidlách.

Biznis pravidlá sa používajú na rozhodovanie v rámci biznis procesov. Základným princípom BRMS riešení je extrakcia biznis logiky z rôznorodých miest výskytu týchto pravidiel do centrálneho úložiska tzv. Rule Repository. Tým je umožnený nezávislý vývoj a editácia pravidiel od zvyšku informačného systému, alebo aplikácie. Vďaka takémuto prístupu môžu byť uskutočňované zmeny nad aplikačnou logikou, resp. rozhodovacími procesmi prehľadne, efektívne a nezávisle od zvyšku aplikácie.

Ako uvádza [39] prístup Case Base Reasoning (CBR – vo voľnom preklade ako vyvodzovanie založené na prípadoch alebo ako prípadové usudzovanie) sa momentálne vyskytuje omnoho častejšie ako tradičný prístup založený na pravidlách. Je to najmä kvôli intuitívnej podstate prípadov, ktoré slúžia ako vhodná reprezentácia znalostí. Tieto prípady môžu byť následne použité na riešenie aktuálneho problému, ktorý je podobný ako niektoré prípady, ktoré už boli vyriešené a uložené pre ich následné znovu použitie práve v takýchto situáciách. CBR je takisto alternatívnym riešením k problému zúženého

hrdla prevzatia znalostí, pretože je vo všeobecnosti jednoduchšie čerpať znalosti z historických záznamov ako modelovať správanie danej domény. Existujú oblasti ako napríklad medicína, krízový manažment, help-desk alebo investovanie na akciovom trhu, kde sú zaznamenávané prípady z minulosti a tie sú opätovne použité pri riešení nových problémov.

Podľa [40] existuje rozdiel medzi CBR a ZS. Znalostné systémy využívajú v rozhodovacích procesoch pravidlá. Znalostný inžinier spolupracuje s doménovým expertom na odvodení pravidiel, ktoré budú použité pri riešení problému. Naproti tomu CBR vyhľadáva podobnosti medzi aktuálnymi potrebami a predchádzajúcimi prípadmi, ktoré sa zaoberali podobným problémom a už pre ne existuje riešenie. Na druhej strane pravidlá a prípady vôbec nie sú natoľko odlišné ako by sa mohlo zdať. Väčšina pravidiel, ktoré využívajú experti sú odvodené z predchádzajúcich skúseností – prípadov.

[39] prezentujú myšlienku využitia CBR v spojení s ontológiou na tvorbu tzv. *knowledge-intensive CBR systému* (KI-CBR – vo voľnom preklade ako „na znalostiach závislé CBR systémy“), kde sú jednoduché CBR prípady obohatené o explicitne definované doménové znalosti. Explicitné definovanie doménových znalostí umožňuje CBR systému „uvažovať“ a vyvodzovať závery omnoho flexibilnejšie (s ohľadom na samotný obsah znalostí) ako keby boli znalosti ukladané len na základe preddefinovaných podobností a váh relevantnosti v CBR prípadoch. Ďalšou výhodou tohto prístupu je znovu použiteľnosť znalostí v systémoch využívajúcich podobnú doménu znalostí.

1.6 Záver kapitoly

Väčšina informačných systémov nezlyháva z technických dôvodov, ale preto, že neriešia skutočné potreby zákazníkov. Nedostatočné upriamenie pozornosti na analytické etapy reprezentované v MDA rovinou CIM iba odsúva skutočné problémy na neskôr, do implementácie a údržby systémov, kedy sú už nápravy chýb rádovo nákladnejšie.

Organizácie potrebujú využívať informačné technológie na podporu špecifických procesov. Vytvoriť informačnú podporu pre takéto špecifické procesy často krát nie je možné získať nakonfigurovaním existujúceho softvérového balíka. V takýchto situáciách je systém, či jeho súčasti nutné vytvoriť na mieru, priamo s ohľadom na požiadavky používateľov. Špecifické procesy vyžadujú logické a efektívne prepojenie dát,

informácií a znalostí prostredníctvom implementácie riadenia znalostí resp. znalostného manažmentu.

Implementácia znalostného manažmentu z pohľadu modelovania vývoja IS je predovšetkým modelovanie roviny CIM. Ak všeobecne tvorba modelov pri vývoji IS zabezpečuje:

- systémový prístup,
- komplexnosť,
- prehľadnosť,
- spôsob vizuálneho vyjadrenia požiadaviek/kritérií IS,
- spôsob zachytenia životného cyklu vývoja IS,

potom pre implementáciu znalostného manažmentu je model CIM modelom najvyššej abstrakcie vo vývoji znalostných systémov.

Modelovanie v CIM rovine MDA a následná transformácia do roviny PIM dáva odpoveď na problémové otázky tvorby informačných systémov, ktoré sú nasledovné:

a) Ako zachytiť funkcionality vytváraného informačného systému?

b) Ako kritéria funkcionalít preniesť priamo do návrhu systému?

c) Ako zabezpečiť adaptívnu zmenu kritérií funkcionalít?

Doterajšie riešenia modelovania CIM roviny v jazyku BPMN a následná transformácia do roviny PIM, uvedené vyššie, v mnohých prípadoch nie sú vhodné pre implementáciu znalostného manažmentu, pretože nedokážu zachytiť dostatočne sémantiku vykonávaných činností. Občas vznikajú nesprávne interpretácie jednotlivých BPMN diagramov, keď sa zamení význam plaveckých dráh s bazénmi (ak je súbor viacerých bazénov považovaný za plavecké dráhy a naopak), tak ako je to uvedené na [41]. Toto však súvisí skôr so samotným modelovaním ako so zachytením sémantiky, ktorá je ale dôsledkom nesprávneho modelovania alebo úsudku následne aj chybne interpretovaná.

Modelovací jazyk BPMN (*Business Process Modelling Notation*) v modelom riadenej architektúre nie primárne koncipovaný na modelovanie znalostí, a nie je možné overiť sémantickú správnosť vytvorených modelov. Rovnako je problém zachovať sémantiku pri transformácií medzi jednotlivými úrovňami. Podporu modelovania znalostí pri vývoji znalostných IS by v tomto smere mohli poskytnúť ontológie. Preto sa ďalej v riešení zameriame na využitie ontológií pre vývoj znalostných IS.

Porovnaním charakteristík ODS (*Ontology Driven Systems*) a OES (*Ontology Enhanced Systems*) zistíme, že ontológia je väčším prínosom v ontologicky riadených systémoch. Pri ich vývoji je ontológia využitá od počiatočnej fázy a ontológiou je riešený komplexnejší problém v porovnaní s riešením OES, kde je ontológia iba doplnkom konvenčného postupu vývoja IS alebo doplnkom existujúceho IS. Aj keď vytváranie architektúry ODS vyžaduje oveľa viac práce ako návrh ontológie OES, správne navrhnutý architektonický rámec pre vývoj ODS môže eliminovať chyby spôsobené rôznymi typmi modelovacích jazykov v rôznych architektonických pohľadoch a transformáciami medzi nimi.

Z predchádzajúcich podkapitol vyplýva, že ontológie majú svoje miesto v rôznych typoch a fázach vývoja IS podľa paradigmy MDA. Napríklad:

- Využitie ontológií v optimalizačných úlohách - kapitola 2.7 vedie na využitie v rôznych inovatívnych riešeniach problémov optimalizácie v rovine CIM modelom riadenej architektúry.
- Ukladanie znalostí vyňatých z textu a následne uložených priamo do ontológie spracované v kapitole 2.5 zlepšuje transformáciu zberu informácií do modelov vývoja IS a súvisí nielen s rovinou CIM modelom riadenej architektúry.
- Vytváranie objektovo orientovaných modelov využitím ontológie pri návrhu IS v kapitole 2.6, vedie na nový prístup modelovania v rovine PIM.
- Modelovanie zodpovedností pomocou ontológie v kapitole 2.4 opäť súvisí s vrstvou CIM, ktorá reprezentuje a modeluje zodpovednosti v IS.

- Modelovanie požiadaviek na IS popísané v kapitole 2.3 sa prekrýva s problematikou vrstvy CIM. V tomto prípade sú požiadavky zachytené pomocou ontológie detailnejšie spracované ako vo vrstve CIM.
- Využitie ontológie pri tvorbe enterprise architektúry spracované v kapitole 2.2 – zachytenie biznis procesov a biznis pravidiel s dodatočným sémantickým významom dokáže poskytnúť vyšší stupeň interoperability medzi ľuďmi a systémami ako to dokáže samotná vrstva CIM.

Z analyzovaných publikácií môžeme konštatovať, že ontológia ako modelovací jazyk je použitá na modelovanie roviny CIM, ktorá tvorí model biznis architektúry, a následnej transformácie do PIM roviny, ktorá predstavuje základ softvérovej architektúry.

Rovina CIM v modelom riadenej architektúre vyjadruje model implementačného prostredia IS. V modernej informačnej spoločnosti, bez ohľadu na typ implementačného prostredia, vyjadruje rovina CIM modely informácií a dát, ich efektívne spracovanie, prepájanie a transformovanie na znalosti. Všetky činnosti všetkých podnikov a inštitúcií sú založené na zbere veľkého objemu informácií, ktoré sú ukladané ako dáta a zároveň sú potrebné ako znalosti pre prevádzku, riadenia a inovácie. Táto skutočnosť tak posúva ontológie do popredia v oblasti tvorby IS, predovšetkým znalostných IS.

Tým sa stáva využitie ontológie, ako nového prístupu pri vývoji informačných systémov, našou výskumnou výzvou.

Niekedy dochádza k prelínaniu pojmov *ontologické* a *znalostné* systémy. Keď sa jedná o ontologický informačný systém, môže byť rozdelený na 2 typy tak, ako bolo uvedené v kapitole 1.2.7:

- Ontologicky riadené systémy (Ontology Driven Systems - ODS)
- Systémy rozšírené o ontológiu (Ontology Enhanced Systems - OES)

Dôležité však je, že súčasťou ontologického systému musí byť ontológia. Ako teda spolu súvisia ontologické a znalostné systémy? Ontologické systémy môžu byť považované za triedu znalostných systémov, pretože ontológia sa v nich využíva na reprezentáciu znalostí. Pre doplnenie možno uviezť, že znalostný systém nie je vytváraný

kvôli ontológii ale kvôli potrebe manažmentu znalostí, ktoré nemusia byť reprezentované ontológiou. Ontologický systém býva vytváraný za účelom práce so znalosťami, preto môže byť považovaný za znalostný systém.

S ohľadom na vyššie uvedené tvrdenia možno prehlásiť, že každý ontologický systém je zároveň aj znalostným systémom, ak sú pre vyjadrenie znalostí použité ontológie. Opačne toto tvrdenie ale neplatí. Každý znalostný systém nie je ontologickým systémom.

2 Špecifikácia výskumného problému

Vývoj informačných systémov, spracovaný v kapitole 1, prináša v poslednom období nový pohľad na spoluprácu na všetkých aktérov vývoja, od používateľov, cez analytikov, softvérových inžinierov až po technikov návrhu infraštruktúry a údržby. Od v minulosti používaného princípu Bottom-Up sa prechádza k princípu Top-Down, kde na najvyššej hierarchickej úrovni je používateľ informačného systému (IS). Nedostatkami IS sú dnes len zriedkavo technické príčiny, nedostatky sú v riešení potrieb používateľov, pre ktorých je IS vyvíjaný. Požiadavky používateľov IS sa stávajú prioritnými informáciami pre stanovenie funkcionalít a ich kritérií pre realizáciu IS.

Vývoj informačných systémov smeruje od doteraz vyvíjaných, široko uplatniteľných systémov, k systémom na podporu špecifických procesov, pre úzku skupinu alebo iba jednotlivých používateľov. Takéto systémy vyžadujú personalizované riešenia, pre ktoré používatelia špecifikujú požiadavky nielen na začiatku vývoja, ale aj počas prevádzky IS a požadujú ich implementáciu v čo najkratšom čase.

Personalizáciu riešení IS je možné spojiť s pojmom *enterprise* architektúra, ktorý budeme ďalej používať ako pojem pre zdôraznenie prepojenia biznisu príslušnej organizácie alebo podniku a vyvíjaným informačným systémom. Všeobecnou paradigmou komplexného vývoja IS je v súčasnosti architektúra systému podľa normy ISO/IEC/IEEE 42010:2011, [5] a v nej špecifikovaný architektonický rámec, vytvorený z požadovaných pohľadov na vyvíjaný IS a vzájomných väzieb medzi pohľadmi. Pohľady vytvárajú parciálne architektúry, ktoré sú vytvárané v príslušných modelovacích jazykoch.

Ak pri vývoji IS uplatníme Top-Down prístup začíname riešiť biznis pohľad architektúry systému. Tento pohľad pre špecifické biznis procesy organizácií vyžaduje logické a efektívne prepojenie dát, informácií a znalostí, ktoré je potrebné riadiť. Prepojenia dát, informácií a znalostí je vhodné riešiť implementáciou riadenia znalostí resp. znalostného manažmentu.

Podľa Business Dictionary je znalostný manažment metóda pre zlepšenie výkonnosti biznis procesov. Systémy riadenia znalostí alebo tiež znalostné systémy zahrňujú ľudské zdroje, biznis stratégiu a informačné technológie. Sú založené na dvoch

kritických aktivitách: zber a dokumentácia špeciálnych explicitných a tacitných znalostí a ich diseminácia v organizácii.

Informačné systémy pre podporu znalostného manažmentu sú potom zamerané na podporu znalostných techník na podporu rozhodovania, učenia a konania. Problematika znalostných systémov je široká, rieši rôzne problémy ich implementácie. Od celkového procesu vývoja, zladenie požiadaviek a potrieb používateľov, aplikácie rôznych znalostných metód, integráciu s konvenčnými technológiami, softvérové nástroje vývoja, rozhodovacie mechanizmy, interakciu používateľov, získavanie a reprezentáciu znalostí, jazyky a programovacie prostredie, techniky implementácie znalostí, systémové architektúry,....

Z pohľadu rovín modelom riadenej architektúry patrí znalostný manažment ako metóda do roviny CIM, ktorá v architektonických rámcoch vývoja IS predstavuje Biznis architektúru. Doterajšie riešenia využívali grafické modelovacie notácie, najčastejšie BPMN – *Business Process Modelling Notation*. Modelovanie CIM roviny v jazyku BPMN a dodržanie striktných pravidiel transformácie medzi modelmi CIM a PIM pre modelovanie princípov znalostného manažmentu nedokáže vždy správne vyjadriť sémantiku dát. Ak uvažujeme modelovanie podľa princípov znalostného manažmentu, je to modelovanie informácií, dát a znalostí. Informácie a znalosti patria v architektonickom rámci do biznis architektúry, dáta patria už do softvérovej architektúry, ktorá v MDA predstavuje rovinu PIM.

Ak ontológia dokáže riešiť vzťahy v CIM (informácie a znalosti) a transformácie CIM-PIM (informácie – dáta) mohla by byť vhodnejším nástrojom pre uplatnenie princípov v MDA, pretože nebude potrebné vytvárať transformačné vzťahy medzi CIM a PIM rovinou. Overenie nášho predpokladu je úlohou ďalšieho riešenia.

3 Cieľ dizertačnej práce

Na základe špecifikácie výskumného problému sme stanovili cieľ práce:

Navrhnúť architektonický rámec riešenia informačného systému pre riadenie znalostí v organizácii použitím princípov modelom riadenej architektúry (MDA) a ontológií ako modelovacieho jazyka v rovinách CIM (*Computer Independent Model*) a PIM (*Platform Independent Model*).

Dosiahnutie uvedeného cieľa je rozčlenené do 4 parciálnych cieľov:

1. Použite ontológií pri vývoji IS vedie na novú disciplínu ontologické inžinierstvo. Ontologické inžinierstvo je pre oblasť IS najčastejšie spojené so znalostným inžinierstvom. Ak MDA je princíp vývoja IS, prvý cieľ riešenia je začleniť princípy MDA do postupov ontologického a znalostného inžinierstva.

2. Všeobecný vývoj IS podľa princípov normy ISO/IEC/IEEE 42010:2011 *Systems and Software Engineering* vedie na vytvorenie architektonického rámca, ktorý tvorí metodiku vývoja *sofware-intensive systems*. Vytvoriť architektonický rámec pre vývoj znalostných systémov s použitím ontológií s dôrazom na modelovanie roviny CIM a PIM je druhý parciálny cieľ.

3. Teoretické riešenie je potrebné overiť. Preto aplikovať vytvorený architektonický rámec pre špecifickú doménu s experimentálnym overením analýzy a návrhu IS je tretí parciálny cieľ.

4. Použitie ontológie ako modelovacieho nástroja v MDA môže odstrániť transformácie medzi rovinami CIM a PIM, ktoré sú potrebné pri použití grafických modelovacích jazykoch. Zdôvodniť a potvrdiť využitie ontológií v modelom riadenej architektúre oproti iným modelovacím jazykom je štvrtý parciálny cieľ.

4 Metodika a metódy riešenia

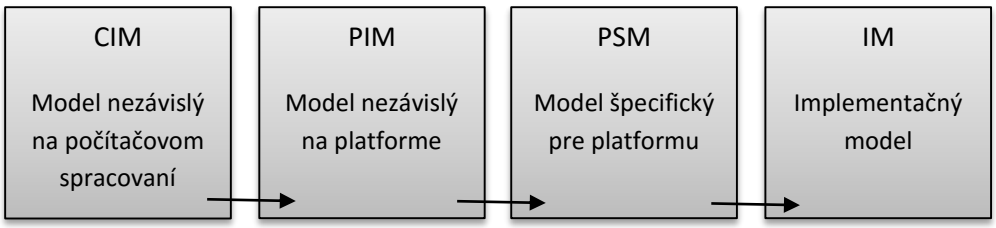
Hlavným objektom skúmania tejto dizertačnej práce je s využitím princípov modelom riadenej architektúry a ontológii navrhnúť architektonický rámec pre znalostné systémy v organizácii. Tento cieľ je rozdelený na 4 parciálne ciele. Metodika riešenia parciálnych cieľov je popísaná v nasledujúcich podkapitolách.

4.1 Modelom riadená architektúra

Prvý parciálny cieľ *Začlenenie princípov MDA do ontologického a znalostného inžinierstva* v rámci metodiky riešenia využíva metódu dedukcie podľa MDA.

Dedukcia odvodzuje nové poznatky z pôvodných premís. Používa zásadu, že pokiaľ sú pôvodné premisy pravdivé, budú pravdivé aj poznatky odvodené z pôvodných premís [42].

MDA určuje štyri typy modelov (úrovní), ktoré v rámci architektúry predstavujú určitý stupeň abstrakcie. Jednotlivé modely, ktoré by mali špecifikovať tieto úrovne abstrakcie nie sú však presne definované. MDA úrovne v poradí v akom sú vytvárané sú zobrazené na obrázku nižšie.



Obrázok 7 - Úrovne abstrakcie MDA

Model nezávislý na počítačovom spracovaní - *CIM* nezobrazuje detaily o konštrukcii IS ale špecifikuje aktivity, ktoré sa v prostredí nasadenia IS vykonávajú. Tento model využívajú biznis analytici, doménoví experti a aj samotní požívatelia. Popisuje architektúru IS, ale skrýva podrobnosti o použití konkrétnej technológie. *Platformovo nezávislý model* - *PIM* vytvára špecifikáciu pre služby IS bez informácii o technických, platformovo závislých detailoch. PIM model obyčajne vytvára systémový analytik. *Model špecifický pre platformu – PSM* je úroveň špecifická z pohľadu konkrétnej platformy. V závislosti od jeho účelu môže obsahovať aj podrobné technické

detaily a takisto môže poskytovať informácie potrebné na implementáciu IS. *Implementačný model – IM* predstavuje zdrojový kód (v danom programovacom jazyku) informačného systému [17].

4.2 Norma ISO/IEC/IEEE 42010:2011

Druhý parciálny cieľ *Vytvorenie architektonického rámca pre vývoj znalostných systémov s použitím ontológií s dôrazom na modelovanie roviny CIM a PIM* v rámci metodiky riešenia využíva metódu dedukcie podľa normy ISO/IEC/IEEE 42010:2011 v spojení s analýzou prostredia znalostných systémov.

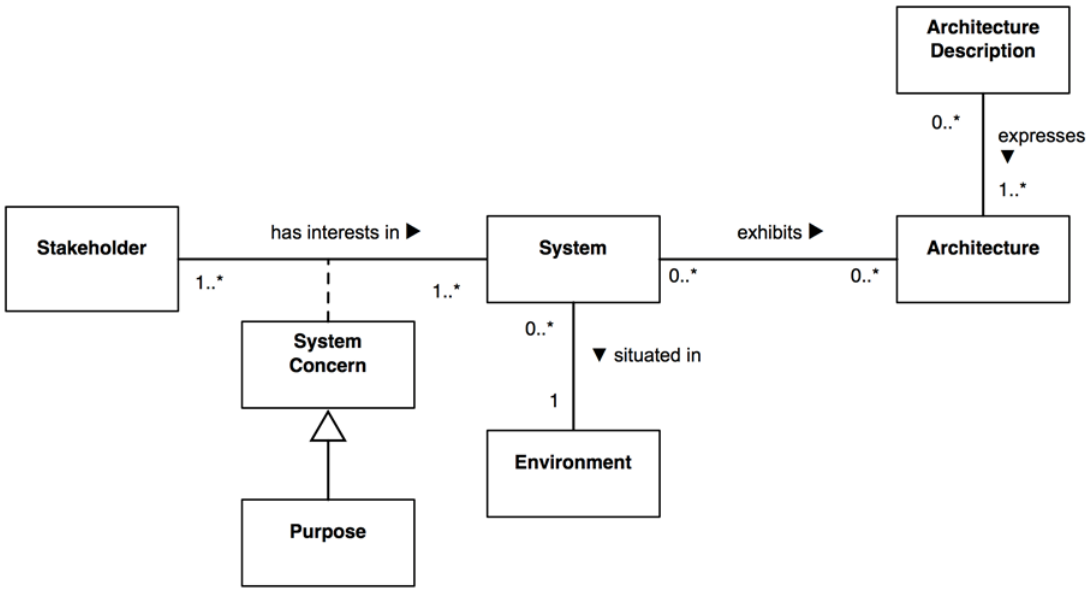
Norma ISO/IEC/IEEE 42010:2011 (ďalej len Štandard) detailne popísaná v [43] je medzinárodný štandard s názvom: *Systémový a softvérový inžiniering*. Tento Štandard sa zameriava na tvorbu, analýzu a udržiavanie architektúr systémov s využitím popisov architektúry.

Norma 42010 odráža súčasný konsenzus osvedčených postupov systémovo-softvérovej komunity pre popis architektúry. Štandard definuje štyri prípady konformity:

1. Popis architektúry.
2. Architektonický rámec.
3. Jazyk popisu architektúry.
4. Hľadisko architektúry.

4.2.1 Popis architektúry

Štandard definuje architektúru ako: *Základné pojmy a vlastnosti systému v jeho prostredí, zakotvené v jeho prvkoch, vzťahoch na základe zásad jeho návrhu a vývoja.* Popis architektúry vyjadruje samotnú architektúru a špecifikuje požiadavky na túto architektúru. Popis architektúry podľa Obrázok 8 - Diagram popisu architektúry - prevzaté z znázorňuje Obrázok 8:



Obrázok 8 - Diagram popisu architektúry - prevzaté z [43]

Systém **(System)** je zasadený do prostredia **(Environment)**, v ktorom môžu existovať aj iné systémy. Štandard definuje systém ako zástupný znak pre podnik, systém systémov, produktové rady, služby, subsystémy alebo softvér.

Zúčastnené strany **(Stakeholders)** sa zaujímajú o systém. Patria sem používatelia systému, systémoví operátori, držitelia systému, vlastníci systému, dodávatelia systému, vývojári a budovatelia systému a takisto správcovia a údržba systému.

Záujmy zainteresovaných strán sú nazývané, v preklade nie veľmi príznačne – obavy **(System Concerns),** preto ponecháme označenie *záujmy*. Mali by tu byť zahrnuté účely systému, vhodnosť architektúry pre dosiahnutie účelov/cieľov systému, realizovateľnosť tvorby a nasadenia systému, možné riziká a dopady systému na zainteresované strany počas jeho životného cyklu, udržateľnosť na vylepšovanie systému.

Cieľ systému **(Purpose)** je všeobecný druh záujmu **(System Concern)**. Systém má svoju architektúru **(Architecture)**. Popis architektúry **(Architecture Description)** sa používa na vyjadrenie architektúry systému.

4.2.2 Architektonický rámec

Architektonický rámec je definovaný ako konvencie, zásady a postupy pre architektúry v špecifickej doménovej oblasti alebo v spoločenstve zainteresovaných strán. Podľa Štandardu 42010 musí tento architektonický rámec špecifikovať:

- Identifikáciu zainteresovaných strán.
- Identifikáciu záujmov, ktoré majú tieto zainteresované strany.
- Architektúru hľadísk, ktoré ohraničujú záujmy.
- Pravidlá integrácie hľadísk.

4.2.3 Popis jazyka architektúry

Popis jazyka architektúry je akákoľvek forma vyjadrenia na popis architektúry. Je potrebné aby jazyk špecifikoval identifikáciu záujmov a zainteresované strany s danými záujmami, modely vyjadrené v popisnom jazyku architektúry, s nimi súvisiace architektonické hľadiská a k nim korešpondujúce pravidlá.

4.2.4 Architektonické hľadisko

Hľadisko môže byť špecifikované nezávisle od popisu architektúry, architektonického rámca alebo jazyka popisu architektúry.

Každé architektonické hľadisko určujú:

- Záujmy ohraničené týmto hľadiskom.
- Zúčastnené strany zainteresované v tomto hľadisku.
- Druhy modelov použité v tomto hľadisku – druh modelu zachytáva konvencie pre typ modelovania. Každý model, ktorý je súčasťou hľadiska definuje jazyky, notácie, konvencie, modelovacie techniky, analytické metódy a prípadne iné operácie potrebné pre model daného druhu.

4.3 Aplikovanie architektonického rámca

Tretí parciálny cieľ *Aplikovanie vytvoreného architektonického rámca pre špecifickú doménu s experimentálnym overením analýzy a návrhu IS.* Na dosiahnutie tohto parciálneho cieľa je použitá metóda analýzy, pretože najskôr je potrebné analyzovať implementačné prostredie, na ktorom bude demonštrované použitie vytvoreného architektonického rámca. Ďalej je použitá metóda dedukcie podľa tohto rámca ale aj metóda komparácie pri určovaní spôsobu ukladania znalostí.

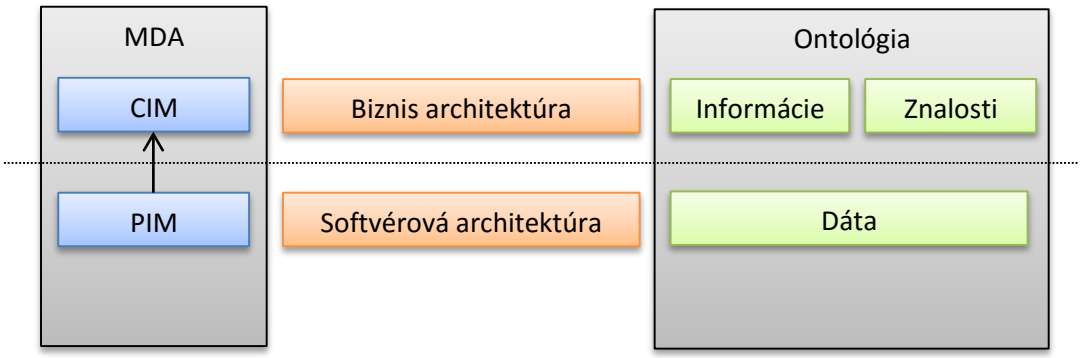
5 Riešenie

Nasledujúce podkapitoly prezentujú riešenie parciálnych cieľov tejto dizertačnej práce, ktoré vychádzajú z metód popísaných v predchádzajúcej kapitole.

5.1 MDA v ontologickom a znalostnom inžinierstve

Ontológia tvorí základ ontologického aj znalostného (ak je vybudované na ontológii) inžinierstva. Princípy MDA v súvislosti s ontológiou boli zhrnuté v kapitole 2, v podkapitole s názvom *Ontológie a MDA*.

Vrstva CIM – *Computer Independent Model* v MDA vyjadruje popis biznis architektúry a vrstva PIM – *Platform Independent Model* popis softvérovej architektúry. Naproti tomu ontológia zachytáva dáta informácie a znalosti. Určite existuje množstvo pohľadov, pomocou ktorých by sa dali vrstvy CIM/PIM prepojiť s ontológiou. Jeden takýto pohľad znázorňuje Obrázok 9. V ideálnom prípade by mohli byť vrstvy CIM a PIM reprezentované ontológiou, ktorá by mohla zjednodušiť proces transformácie, prípadne potrebu transformácie úplne odstrániť.



Obrázok 9 - Previazanie MDA s ontológiou

Spomenutým spôsobom je možné využiť princípy MDA v rámci ontologického a znalostného inžinierstva. Ontológie sú často súčasťou znalostných systémov, avšak tvorba znalostných systémov z architektonického hľadiska nie je triviálna záležitosť.

Nasledujúca podkapitola sa venuje tvorbe architektonického rámca pre vývoj znalostných systémov s použitím ontológií s dôrazom na modelovanie roviny CIM a PIM.

5.2 Architektonický rámec znalostných systémov

Mať dobre navrhnutú architektúru je rozhodujúce pre úspech systému alebo podniku. Za účelom správneho návrhu systémovej architektúry je často využívaný architektonický rámec. Architektonický rámec možno podľa [44] charakterizovať ako opornú štruktúru/predpis, ktorá definuje architektonické artefakty, popisuje ako sú tieto artefakty prepojené medzi sebou a popisuje všeobecné definície toho, ako by tieto artefakty mohli vyzerať.

5.2.1 Zainteresované strany v znalostnom systéme

Okruh osôb, ktoré sú ovplyvnené informačným systémom nie je obmedzený iba na osoby, ktoré ho priamo používajú – používateľov. Informačný systém nie je len o jeho samotnom využívaní – musí byť vyvinutý, testovaný, obsluhovaný, udržovaný a opravovaný, vylepšovaný a samozrejme zaplatený. Každá z týchto činností zahŕňa množstvo ľudí, z ktorých sa profilujú zainteresované strany, ktoré majú vlastné záujmy, požiadavky a potreby voči danému informačnému systému. Autori knihy *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives* [45] určili zainteresované strany (skupiny) nasledovne:

- **Nadobúdatelia** – dohliadajú na obstaranie systému
- **Hodnotitelia** – dohliadajú aby bol systém v súlade so štandardami a právnymi predpismi.
- **Komunikátori** – vysvetľujú systém iným zainteresovaným cez systémovú dokumentáciu a výukové materiály.
- **Vývojári** – vytvárajú a nasadzujú systém na základe špecifikácie (prípadne aj vedú tím(y) vývojárov).
- **Správcovia** – riadia systém ako náhle je v prevádzke.
- **Produkční inžinieri** – návrh, nasadenie a správa hardvérových prostredí, v ktorých bude systém vyvíjaný, testovaný a nasadený.

- **Dodávatelia** – vytvárajú/dodávajú hardvér, softvér, či infraštruktúru, na ktorej bude systém fungovať.
- **Pracovníci podpory** – poskytujú podporu používateľom systému po jeho nasadení.
- **Systémoví administrátori** – sú zodpovední za beh systému ako náhle bol nasadený.
- **Testeri** – testujú systém aby overili jeho použiteľnosť.
- **Používatelia** – definujú funkcionality systému a používajú ho.

Autori [45] tvrdia, že mnohé z projektov, ktoré sa venujú vývoju informačných systémov zahŕňajú väčšinu (ak nie všetky) zo zainteresovaných strán uvedených vyššie, avšak ich dôležitosť je niekedy relatívna a líši sa od konkrétneho projektu. Príliš veľká rôznorodosť zainteresovaných strán nie je neobvyklá, no v istých prípadoch môže zbytočne spomaliť napredovanie projektu bez dodatočnej pridanej hodnoty.

Keďže znalostné systémy sú podoblasť informačných systémov, zainteresované strany znalostných systémov môžu byť veľmi podobné. Avšak oblasť získavania a uchovávania znalostí v znalostných systémoch je úzko previazaná, na jednej strane s reálnymi znalosťami a skúsenosťami doménových expertov a na strane druhej s schopnosťami znalostných inžinierov, ktorí dokážu znalosti vložiť do znalostného systému. Preto pre oblasť znalostného manažmentu vystupujú ďalšie dve zainteresované strany – doménový expert a znalostný inžinier.

Ako bolo uvedené už v Kap. 2 všeobecná problematika znalostných systémov je veľmi rozsiahla, a preto je možné, že niektoré zainteresované strany sa pre konkrétne problémové úlohy môžu líšiť.

Podľa vyššie uvedených informácií môžeme pre znalostné systémy špecifikovať minimálne nasledovné zainteresované strany:

- Používatelia systému.

- Znalostní inžinieri.
- Doménoví experti.
- Vývojári systému.
- Systémoví administrátori.
- Dodávatelia systému.
- Systémová údržba.

5.2.2 Záujmy zainteresovaných strán v znalostnom systéme

Každá zo zainteresovaných strán má svoje záujmy, ktoré vymedzujú jej pole pôsobnosti v danom systéme.

Používatelia systému

Používatelia znalostného systému očakávajú (okrem zaužívaných požiadaviek, akými sú robustnosť, prehľadnosť, dostupnosť, bezpečnosť a príjemné používateľské prostredie) najmä distribúciu a uchovávanie znalostí tak, aby boli dostupné a použiteľné v takej forme, ktorá dokáže zabezpečiť zefektívnenie práce a zlepšenie jej výsledkov. Interpretácia znalostí do takejto formy nie je triviálnou úlohou, preto je nevyhnutná komunikácia s ďalšími zainteresovanými stranami.

Znalostní inžinieri

Znalostní inžinieri vystupujú v oblasti znalostného inžinierstva (časť Znalostné inžinierstvo) a sú dôležitou súčasťou znalostného systému. Práve oni dokážu naplniť systém požadovanými znalosťami. Ich záujmom je, aby systém poskytoval unifikovaný spôsob zapisovania znalostí do systému, najlepšie aj s kontrolou vstupov a logických chýb, ktoré by sa prejavili neskôr na výsledkoch poskytnutých koncovým používateľom. Používatelia, tieto chyby následne nedokážu odhaliť. V ojedinelých prípadoch môžu byť niektorí používatelia zároveň znalostnými inžiniermi.

Doménoví experti

Sú to experti z praxe, ktorí disponujú znalosťami z danej doménovej oblasti. Ich vedomosti sú významným zdrojom znalostí pre znalostný systém. Spolupracujú so znalostnými inžiniermi pri napĺňaní znalostného systému. Nie je neobvyklé, keď sú doménoví experti zároveň používateľmi znalostného systému. Doménoví experti majú záujem na tom, aby systém reprezentoval ich znalosti čo najvierohodnejšie bez straty súvislosti a významu. Takisto tu vstupuje aj morálna otázka ohľadne vlastníctva vedomostí doménových expertov. V znalostnom systéme by za žiadnych okolností nemalo dôjsť k odcudzeniu znalostí expertov – práve naopak integrácia znalostí do väčších a ľahšie prístupnejších celkov má pomôcť doménovým expertom pri nadobúdaní nových znalostí, ktoré im umožnia posunúť danú doménovú oblasť na novú úroveň.

Vývojári systému

Ich úlohou je vyvinúť znalostný systém podľa požiadaviek používateľov. Ich záujmom je presná a jasná špecifikácia požiadaviek na systém.

Systémoví administrátori

Podobne ako pri požívateľoch aj administrátori majú záujem na tom, aby bol systém z administrátorského hľadiska robustný, bezpečný ale aj ľahko spravovateľný.

Systémová údržba

Niekedy nie je možné jednoznačne odlíšiť administráciu systému od údržby, avšak v prípade znalostných systémov sa údržbou myslí nasadzovanie nových verzií, aktualizácie a riešenie problémov súvisiacich s riadením znalostí, ktoré vzniknú v systéme. Údržbu často vykonávajú osoby, ktoré poznajú systém detailne po technickej stránke a to sú v tomto prípade práve samotní vývojári, prípadne osoby, ktoré boli do takejto údržby zaškolené.

Dodávatelia

Ich záujmom je vytvoriť/dodať hardvér, softvér, či ďalšiu infraštruktúru, na ktorej bude systém prevádzkovaný.

5.2.3 Architektúra hľadísk v znalostnom systéme

Názory na hľadiská v systémovej architektúre sa rôznia. Niekedy je dokonca výraz *hľadisko* zamieňaný s výrazom *pohľad*. Rozdiel medzi dvomi výrazmi možno zadefinovať tak, že *pohľad* vyjadruje *na čo* sa pozorovateľ pozerá a *hľadisko* reprezentuje *odkiaľ* sa pozerá (tzn. pozícia konkrétneho pozorovateľa). Pohľad môže mať viacero hľadísk (pozorovateľov).

Existuje množstvo rozdelení hľadísk v rámci softvérovej architektúry IS. Zdroj [46] rozdeľuje hľadiská do viacerých skupín, napríklad:

- Hľadiská logického návrhu
    - Hľadisko komponentov,
    - Hľadisko interakcií komponentov,
    - Hľadisko stavov komponentov,
    - Hľadisko vrstiev a subsystémov,
    - Hľadisko logických dát,
    - Hľadisko závislostí subsystémov.
- Fyzické hľadiská
    - Hľadisko nasadenia,
    - Hľadisko fyzických dát,
    - Procesné hľadisko,
    - Hľadisko procesných stavov.

Iné rozdelenie uvažujú [47]:

- Hľadisko založené na biznis doméne – zahŕňa koncepty biznis domény a ich vzťahy.

- Hľadisko založené na vzoroch – identifikuje architektonické elementy, ktoré by mali odpovedať zvoleným vzorom.

- Kohézne (súdržné) hľadisko – identifikuje súbor príbuzných architektonických elementov so silnými závislosťami.

Hľadiská v systémovej architektúre sú podľa [45] rozdelené nasledovne:

- Funkcionálne hľadisko – popisuje funkčné prvky systému, ich zodpovednosti, rozhranie a primárne interakcie.

- Informačné hľadisko – popisuje spôsob, akým sú ukladané, spracované a distribuované informácie.

- Simultánne hľadisko – popisuje súbežnosť systému tak, aby jeho časti boli koordinované a kontrolované súbežne.

- Hľadisko vývoja – popisuje architektúru, ktorá podporuje proces softvérového vývoja.

- Hľadisko nasadenia - popisuje prostredie, do ktorého bude systém nasadený (hardvér, softvér tretích strán, kompatibilita, kapacita, obmedzenia, ...)

- Operačné hľadisko – popisuje ako bude systém ovládaný, administrovaný a udržovaný po jeho nasadení.

Z uvedených príkladov vyplýva, že poňatie hľadísk nie je možné jednoznačne generalizovať. Mnoho systémových architektov sa snaží hľadiská prispôsobiť podľa konkrétnej problémovej úlohy.

Autor v tejto práci vychádza z rozdelenia podľa [45], pretože tento prístup poskytuje ucelený pohľad na problematiku hľadísk. Aj napriek tomu, že sa nejedná o štandard, ide o je prístup uznávaný mnohými, o čom svedčia pozitívne recenzie svetovo uznávanými inštitúciami ale aj stovky citácií vo vedecko-odborných publikáciách, ako napríklad knihy od Van Vliet, Hans, Hans Van Vliet, and J. C. Van Vliet. *Software*

engineering: principles and practice. Vol. 3. Wiley, 2007 alebo Babar, Muhammad Ali, et al. *Software architecture knowledge management*. Springer Science & Business Media, 2009. a ďalšie.

Spomínané hľadiská podľa [45] majú isté spoločné črty, a preto by bolo možné zjednotiť ich do troch nadradených pohľadov, ktoré sa vyskytujú v každom informačnom systéme, a síce:

1. Biznis pohľad
    - Funkcionálne hľadisko.
    - Simultánne hľadisko.
    - Operačné hľadisko.
2. Softvérový pohľad
    - Informačné hľadisko.
    - Hľadisko vývoja.
3. Hardvérový pohľad
    - Hľadisko nasadenia.

Keďže uvedené rozdelenie (pohľady a hľadiská) je definované pre informačné systémy vo všeobecnosti, vyhovuje aj architektúre hľadísk (a aj pohľadov) pre znalostné systémy. Napriek tomu je dôležité poznamenať, že všetko závisí od uhľa pohľadu, a preto by napr. *Hľadisko nasadenia* mohlo figurovať aj v rámci *Softvérového hľadiska*.

Keďže MDA v rámci špecifikovaných úrovní neuvažuje hardvérový pohľad a tvorba všeobecného hardvérového pohľadu by nemala žiadaný prínos pri tvorbe konkrétneho znalostného systému, autor v tejto práci hardvérový pohľad bližšie nešpecifikuje.

5.2.4 Návrh architektonického rámca pre znalostné systémy

Pri návrhu architektonického rámca sú brané do úvahy informácie o znalostných systémoch uvedené v predchádzajúcich častiach. Tak ako už bolo uvedené v časti Znalostné systémy - Z pohľadu vývoja znalostných softvérových systémov možno tvrdiť, že ich princípy sú veľmi podobné so všeobecnými princípmi vývoja IS. Líšia sa po väčšine len v tom, že znalostné systémy musia riešiť nielen spracovanie informácie ale aj *manažment znalostí* – reprezentovanie, ukladanie a získavanie znalostí.

Pri návrhu architektonického rámca je treba zdôrazniť 3 elementy, ktoré sú súčasťou znalostného systému:

1. Dáta, informácie, znalosti.
2. Manažment znalostí.
3. Znalostné inžinierstvo.

Manažment znalostí pracuje s dátami, informáciami a znalosťami, ktoré s použitím znalostného inžinierstva (v tomto prípade je to asistencia znalostného inžiniera) spravuje a transformuje ich na znalosti, ktoré sú ukladané v znalostnej báze. Ďalej manažment znalostí prijíma dotazy, ktoré spracuje a následne v znalostnej báze vyhľadá riešenie daného dotazu a poskytne odpoveď dotazovateľovi (napríklad používateľovi IS).

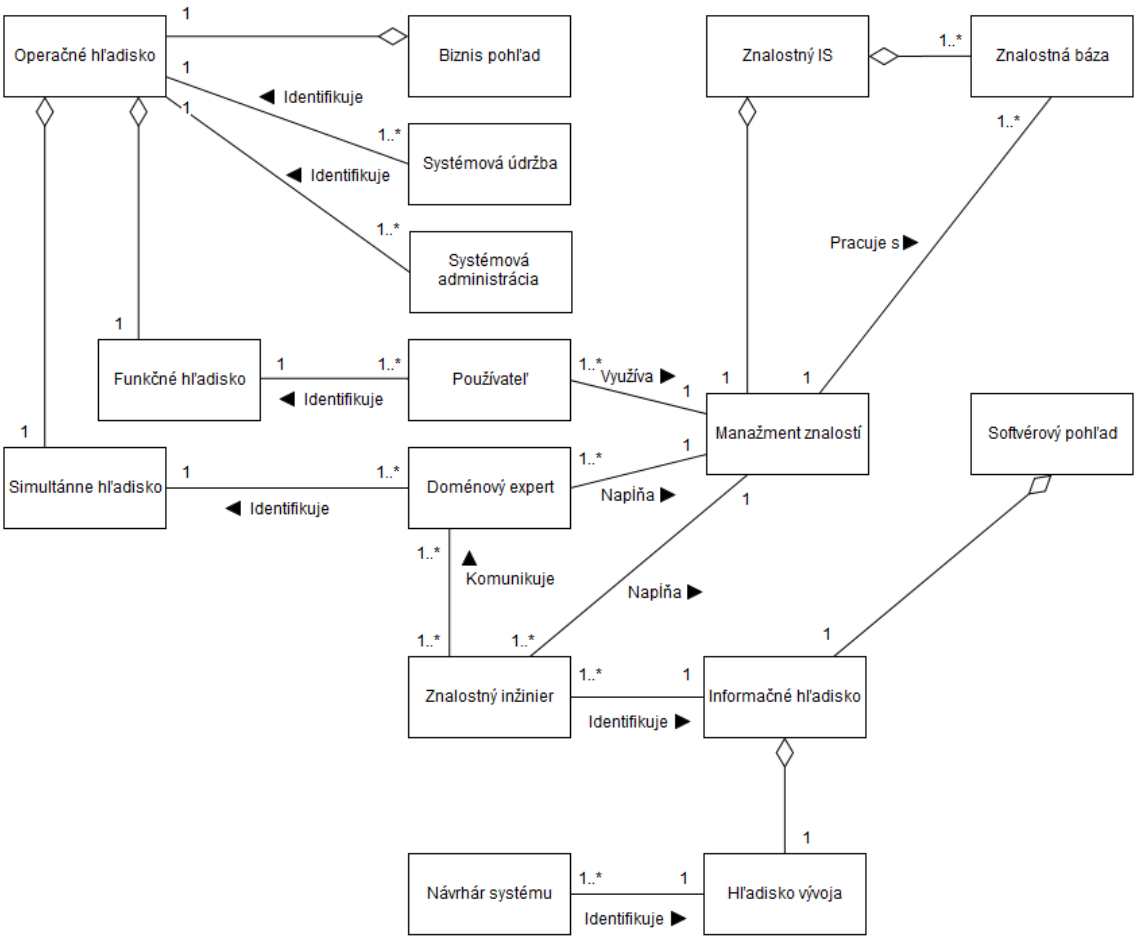
Na základe uvedených informácií o znalostných systémoch ale aj o zainteresovaných stranách v znalostných systémoch, vychádzajúc z rozdelenia hľadísk podľa [45] a ich následného zjednotenia do nadradených pohľadov je možné architektonický rámec znalostných systémov popísať nasledovne:

Biznis pohľad v sebe zahŕňa zainteresované strany – *používateľ* a *doménový expert*, ktoré priamo súvisia s funkcionálnym, simultánnym a operačným hľadiskom, pretože používateľ definuje a neskôr po nasadení systému využíva funkčné prvky systému (*funkčné hľadisko*) a takisto popisuje ako bude systém ovládaný (*operačné hľadisko*). To isté platí aj pre doménového experta, ktorý sa na systém pozerá z pohľadu znalostí, ktoré budú premietnuté do systému a navyše na základe jeho znalostí dokáže

popísať súbežnosť systému (*simultánne hľadisko*). Ďalej biznis pohľad v rámci *operačného hľadiska* zahŕňa *systémovú údržbu* a *systémovú administráciu*.

Softvérový pohľad je tvorený *vývojármi systému* a *znalostným inžinierom*. Vývojári systému riešia *informačné hľadisko* a *hľadisko vývoja*. Znalostný inžinier v *informačnom hľadisku* integruje hľadisko doménového experta v rámci systému.

Hardvérový pohľad obsahuje *hľadisko nasadenia*, ktoré ako už bolo spomínané popisuje prostredie, do ktorého bude systém nasadený. V rámci hardvérového pohľadu bude vytvorené aj prostredie pre tvorbu a ukladanie znalostnej bázy. Ako už bolo spomínané vyššie hardvérový pohľad nie je v tejto práci bližšie špecifikovaný, a preto nie je zahrnutý ani v grafickom návrhu architektonického rámca pre znalostné systémy, ktorý ilustruje Obrázok 10.



Obrázok 10 - Grafický návrh architektonického rámca pre znalostné systémy

Obrázok je znázornený formou konceptuálneho modelu kontextu systému [48] podľa ISO/IEC/IEEE 42010:2011.

Vrstva CIM *Computer Independent Model* je vyjadrená *Biznis pohľadom*. Vrstva PIM – *Platform Independent Model* je reprezentovaná *Softvérovým pohľadom* (ale iba medziach platformovej nezávislosti). Ontológie sú zakomponované v rámci znalostnej bázy resp. v rámci znalostného systému. Týmto spôsobom navrhnutý architektonický zachováva dôraz na modelovanie roviny CIM a PIM a zároveň umožňuje využiť ontológie.

5.3 Využitie vytvoreného architektonického rámca

Teoretický návrh architektonického rámca je potrebné prakticky overiť. Keďže sa jedná o architektonický rámec, ktorý je určený pre znalostné systémy, je vhodné zvoliť taký problém, ktorého vyriešenie požaduje využitie znalostí. Znalostné systémy môžu na prácu so znalosťami využívať ontológiu – v tomto prípade je možné hovoriť o ontologických informačných systémoch.

Ako bolo uvedené v podkapitole 1.2.7 medzi ontologické informačné systémy možno zaradiť aj niektoré systémy určené pre procesné a systémové modelovanie, diagnostiku, podporu rozhodovania či plánovania.

Overenie vytvoreného architektonického rámca bude demonštrované pri návrhu systému pre podporu plánovania procesov priemyselnej výroby.

5.3.1 Analýza systému pre podporu plánovania procesov priemyselnej výroby

Plánovanie procesov je neoddeliteľnou súčasťou priemyselnej výroby ale aj priemyslu ako takého. Vhodné plánovanie výrobných procesov dokáže zabezpečiť nielen plynulú výrobu a dodávku tovarov, ale aj efektívne znížiť náklady, zrýchliť a optimalizovať proces celej výroby. Ambícia minimalizovať chyby v oblasti plánovania donútila priemysel automatizovať (resp. poloautomatizovať) aj samotný návrh a plánovanie výrobných procesov.

5.3.2 Zainteresované strany a ich záujmy

Zainteresované strany v systéme pre podporu plánovania priemyselnej výroby možno rozdeliť do niekoľkých skupín:

- Používatelia systému.
- Znalostní inžinieri.
- Doménoví experti.
- Vývojári systému.
- Systémoví administrátori.
- Dodávatelia systému.
- Systémová údržba.

Ako si možno všimnúť, uvedené skupiny sú analógiou zainteresovaných strán uvedených v podkapitole 5.2.1 Týmto zainteresovaným skupinám prináležia záujmy, ktoré boli predstavené v podkapitole 5.2.2.

5.3.3 Návrh systému pre podporu plánovania priemyselnej výroby

Keďže sa jedná o znalostný systém, v rámci návrhu bude potrebné upriamiť pozornosť na prepojenie systému so znalosťami, ktoré budú použité pri podpore plánovania priemyselnej výroby. Ako už bolo uvedené, znalosti možno uchovávať v akejkoľvek forme. V informačných systémoch sú dáta, informácie a znalosti uchovávané napríklad v textovej podobe, v databázových systémoch alebo v ontológiách.

Uchovávanie znalostí v samotnom systéme pre podporu plánovania priemyselnej výroby sa môže líšiť prípad od prípadu a závisí od konkrétneho návrhu. V rámci overenia vytvoreného architektonického rámca autor zohľadňuje 2 spôsoby uchovávania znalostí:

- Databázový systém
- Ontológia

Ukladanie znalostí s využitím databázového systému

Pod databázovým systémom (DBS) je treba chápať množinu navzájom súvisiacich dát spoločne s programovým vybavením, ktoré umožňuje prístup k dátam.

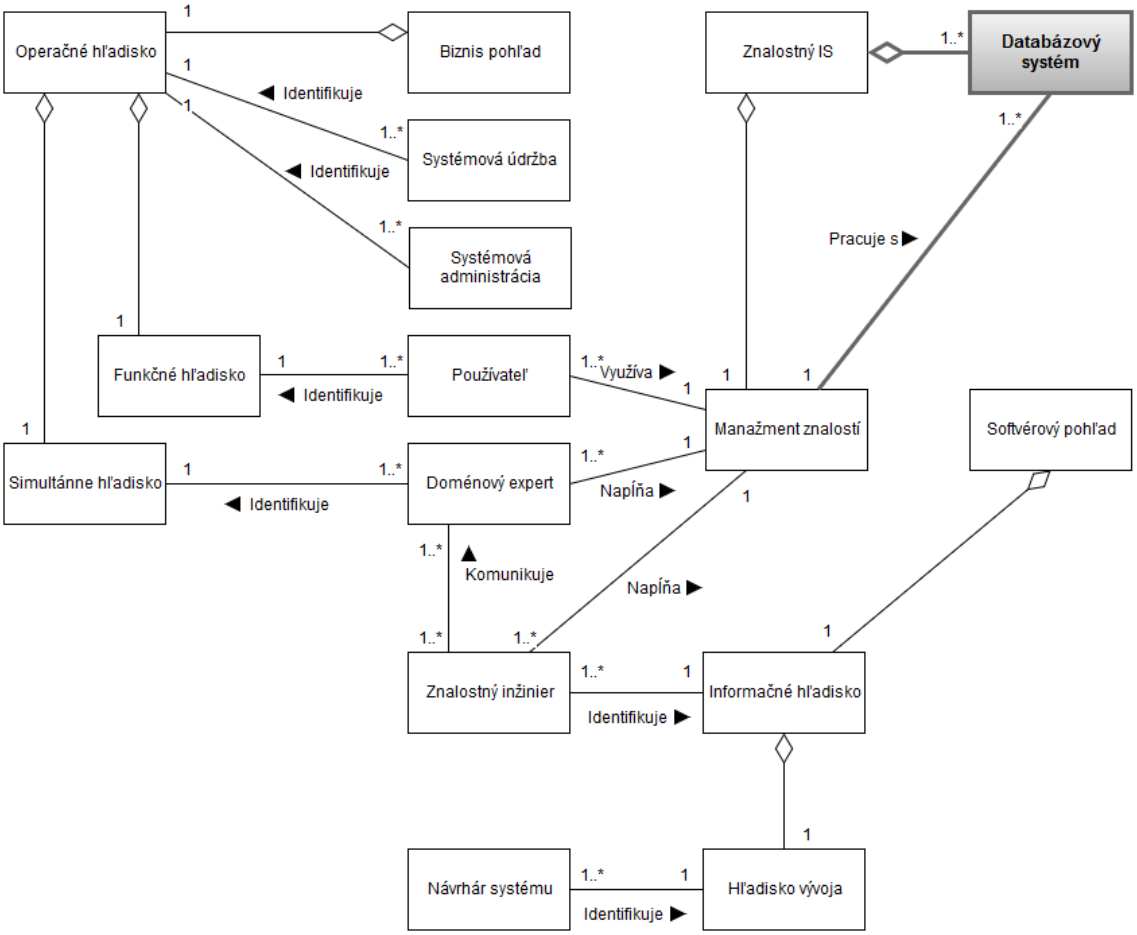
Architektúra databázy predstavuje všeobecný databázový koncept pre vysvetlenie štruktúry databázového systému. Architektúra vychádza z návrhu ANSI/SPARC Study.

Group on Data Base Management Systems, ktorý bol vyvinutý v 70-tych rokoch pri sieťovom databázovom systéme a je platný aj pre súčasné databázové systémy [49].

Spracovanie každej požiadavky vyskytujúcej sa v aplikácii priamo súvisí s architektúrou databázového systému. Systém riadenia bázy dát (ďalej len SRBD) musí byť schopný interpretovať požiadavku, ktorá reprezentuje externý pohľad, s použitím externej schémy, konceptuálnej a internej schémy na konkrétnu požiadavku sprístupnenia, resp. zápisu dát do externej pamäte. SRBD predstavuje množinu programov, zabezpečujúcich manipuláciu s dátami, ochranu dát, paralelné spracovanie, a pod. Počítačové systémy, na ktorých sú prevádzkované databázové systémy je možné rozdeliť na tieto architektúry [49]:

- Centralizované.
- Súborové systémy.
- Klient / server.
- Distribuované.

Pri použití databázového systému pre ukladanie znalostí by mohol architektonický rámec znalostných systémov vyzerať nasledovne:



Obrázok 11 - Grafický návrh architektonického rámca pre znalostné systémy využívajúce databázový systém

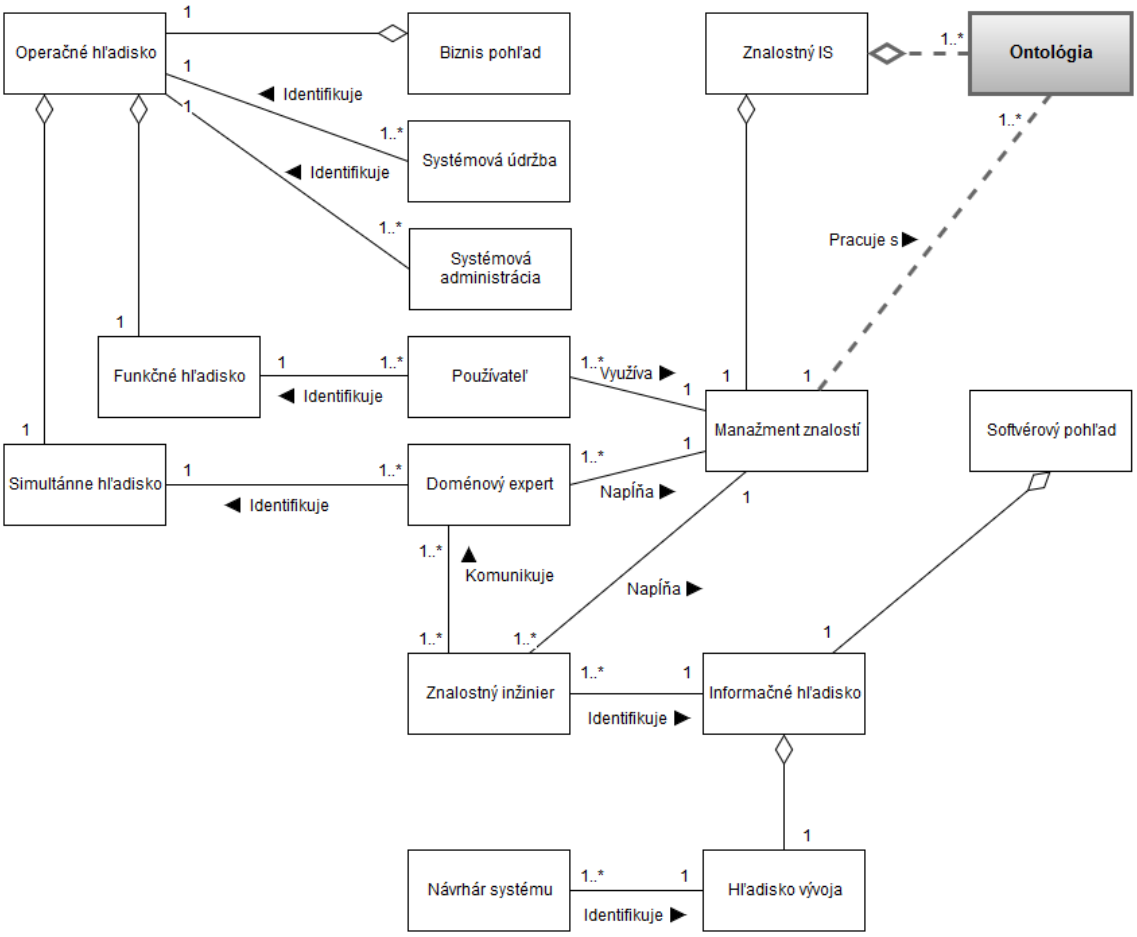
Ako je možné vidieť, znalostná báza bola nahradená databázovým systémom. Výsledná podoba architektonického rámca sa v rôznych prípadoch môže líšiť.

Ukladanie znalostí s využitím ontológie

Druhým spôsobom pre ukladanie znalostí je využitie ontológie. Rôzne možnosti využitia ontológie v rámci IS boli popísané v Kapitole 1.2.

Ontológiu v prostredí, kde sa súčasne nachádza viacero IS v enterprise architektúre je možné použiť na zjednotenie a integráciu dát, informácii a znalostí v organizácii. S ohľadom na tento fakt je potrebné ontológiu navrhnúť a vytvoriť samostatne a následne umožniť jej integráciu s viacerými IS súčasne. Takto vytvorená ontológia by bola napĺňaná cez znalostný systém ale znalosti v nej obsiahnuté by boli dostupné aj pre ďalšie IS v organizácii.

Pri využití ontológie je znalostná báza v návrhu architektonického rámca nahradená ontológiou, ktorá je vyňatá zo znalostného systému, aby mohla komunikovať aj s inými IS v rámci organizácie, tak ako je to znázornené na nasledujúcom obrázku. Vyňatie ontológie je znázornené čiarkovanými čiarami.



Obrázok 12 - Grafický návrh architektonického rámca pre znalostné systémy využívajúce ontológiu

Vyňatie ontológie mimo znalostného systému je iba jeden so spôsobov ako vnímať túto problematiku zo širšej perspektívy ale ontológia môže byť aj priamo integrovaná v znalostnom systéme. To či sa bude jednať o *ontologicky riadený systém* alebo o *systém rozšírený o ontológiu* (ako to bolo spomenuté v podkapitole 1.6) záleží od konkrétneho návrhu.

5.3.4 Výsledné riešenie experimentálneho overenia

Jednotlivé časti analýzy a návrhu uvedené v predchádzajúcich podkapitolách vychádzajú priamo z navrhnutého architektonického rámca, ktorého cieľom je sprehľadniť a uľahčiť etapu návrhu znalostného informačného systému.

Výsledným riešením experimentálneho overenia je desktopová aplikácia systému pre podporu plánovania procesov priemyselnej výroby vytvorená v platforme JavaFX.

Aplikácia poskytuje používateľovi podporu počas plánovania výrobných procesov na základe konfigurácie sledovaných parametrov a možných scenárov konkrétneho výrobného procesu. Medzi sledované parametre patria:

- Výroba konkrétneho druhu produktu alebo jeho časti,
- Počet pracovísk,
- Počet pracovníkov,
- Počet robotov,
- Časové presuny medzi pracoviskami
- Výška nákladov na vykonanie jednotlivých procesných operácii,
- Časová náročnosť jednotlivých procesných operácii,
- Časové verzus nákladové preferencie a ich úspora,
- Množstvo vyrobených kusov vzhľadom k nákladom a k spotrebovanému času.

Využitie aplikácie je ilustrované na príklade procesu výroby priemyselnej vidlice.

Obrázok 13 - Priemyselná vidlica - prebraté z [50]

Vidlica sa skladá z niekoľkých komponentov:

- skrutka s nulovým kolíkom,
- skrutka s fázovým kolíkom,
- telo krytu,
- hlava vidlice ,
- nálepka.

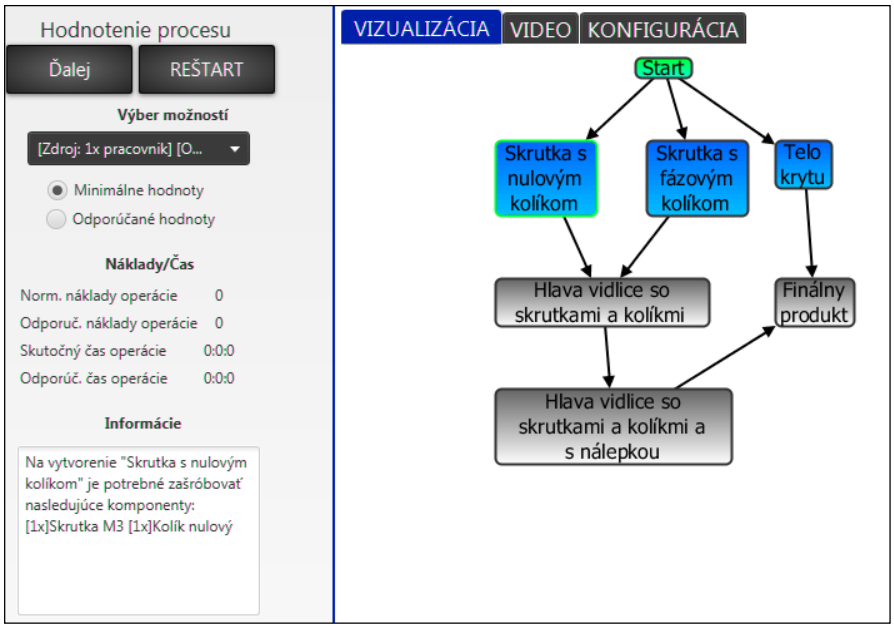
Zmontovanie priemyselnej vidlice podlieha výrobnému procesu, kedy je v danom okamihu možné montovať spolu iba niektoré komponenty. Takýmto spôsobom vznikajú medziprodukty (*hlava vidlice so skrutkami a kolíkmi, hlava vidlice so skrutkami a kolíkmi a nálepkou*). Postupné montovanie komponentov a medziproduktov končí vytvorením finálneho produktu – premyslenej vidlice.

Postupnosť krokov výrobného procesu, obmedzenia, informácie o dostupných zdrojoch, operáciách a ich vzájomná previazanosť, sú znalosti, ktoré musí aplikácia ukladať, spracovať a ponúknuť používateľovi vo forme aktuálne dostupných možností počas plánovania výrobného procesu.

Aplikácia zobrazuje vizualizovaný prechod výrobným procesom, kde používateľ vidí, ktoré aktivity je možné v danom okamihu vykonať (odlíšené modrou farbou –

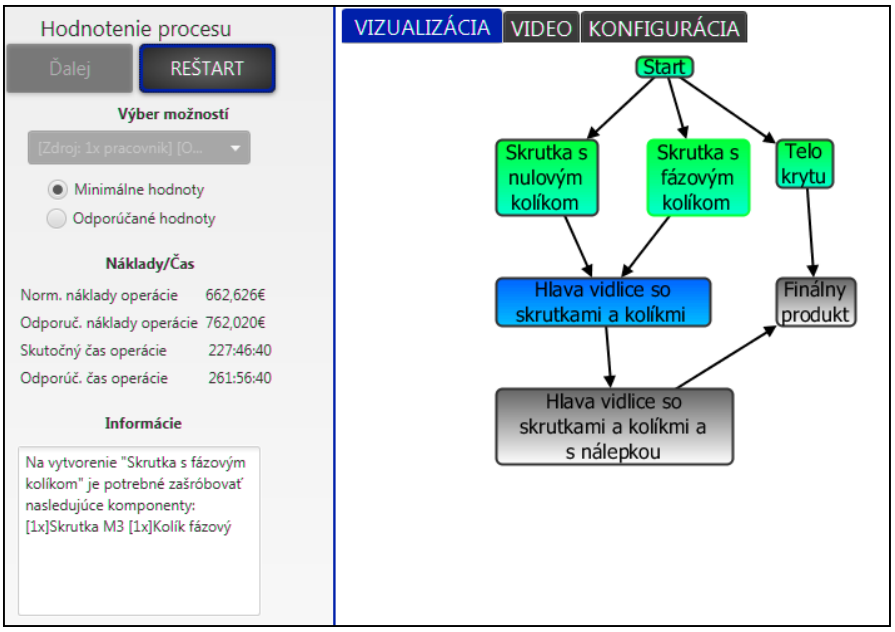
Obrázok 14). V ľavej časti okna aplikácie sa nachádza *Hodnotenie procesu*, ktoré umožňuje používateľovi:

- meniť dostupné zdroje (počty pracovníkov a priemyselných robotov),
- časy spracovania nastaviť na požadované hodnoty,
- vidieť aktuálne vznikajúce náklady a čas operácie,
- vidieť doplňujúce informácie o montáži komponentov.



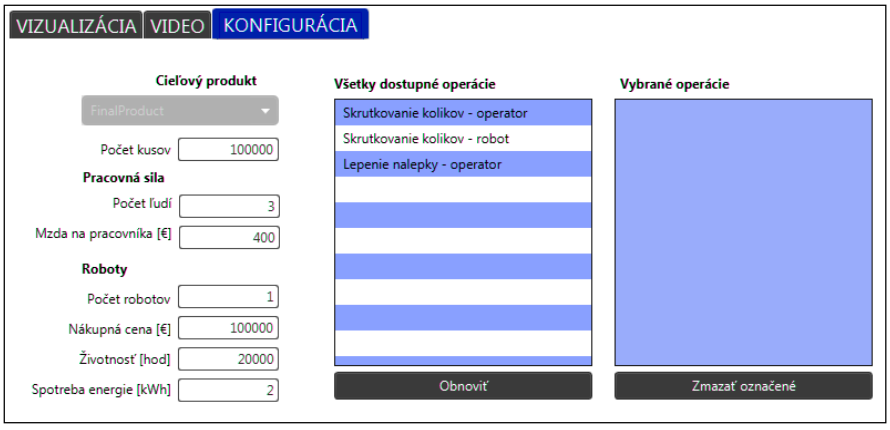
Obrázok 14 - Prechod výrobným procesom 1

Komponenty, ktoré prešli procesom montáže sú následné označené zelenou farbou tak ako je to možné vidieť na Obrázku 15. Pre lepšie pochopenie procesu výroby daného produktu umožňuje karta *Video* zobraziť animovaný náhľad tohto procesu.



Obrázok 15 - Prechod výrobným procesom 2

Karta *Konfigurácia* umožňuje nastaviť výrobné kapacity, náklady a operácie, ktoré sa majú pri danom plánovaní výrobného procesu použiť.



Obrázok 16 - Konfigurácia

Uvedená funkcionalita vytvára z aplikácie ľahko konfigurovateľnú a rýchlo dostupnú podporu plánovania výrobných procesov. Netreba však zabúdať na to, že kvalita aplikácie sa v značnej miere odvíja od kvality znalostí, ktorými aplikácia disponuje. Preto, ako už bolo spomenuté v Kapitole 5, pri tvorbe znalostných systémov netreba zabúdať na doménových expertov, ktorí potrebnými znalosťami disponujú a znalostnými inžiniermi, ktorí tieto znalosti dokážu vložiť do systému.

5.4 Použitie ontológie ako modelovacieho nástroja v MDA

Použitie ontológie ako modelovacieho nástroja v MDA môže odstrániť transformácie medzi rovinami CIM a PIM, ktoré sú potrebné pri použití grafických modelovacích jazykoch. Zjednotenie rovín CIM a PIM vyžaduje zjednotenie aj na úrovni notácie. Preto je dôležité v rámci jednotného ontologického zápisu (modelu) vybrať také prvky, ktoré by dokázali reprezentovať a zachytiť relevantné informácie z oboch rovín.

5.4.1 Existujúce transformačné prístupy

Problematika transformácii jednotlivých úrovní (nie len v rámci MDA) pri návrhu IS je stále aktuálna, a preto sú vytvárané prístupy, ktoré sa snažia tento proces zjednodušiť.

Jedným z takýchto prístupov prezentuje aj práca *Transformácia CIM do PIM v modelom riadenej architektúre MDA* [17], kde je vytvorený polo-automatizovaný postup transformácie z BPMN notácie do notácie UML.

Ďalším prístupom je štandard QVT (Query/View/Transformation), ktorý rieši metamodelové transformácie pomocou QVT jazykov. Tieto jazyky sú:

- Relácie – deklaratívny transformačný jazyk, ktorý špecifikuje vzťahy medzi elementami modelu.
- Jadro - deklaratívny transformačný jazyk, ktorý zjednodušuje jazyk relácii.
- Operatívne mapovania - imperatívny transformačný jazyk, ktorý rozširuje jazyk relácii o imperatívne konštrukty.

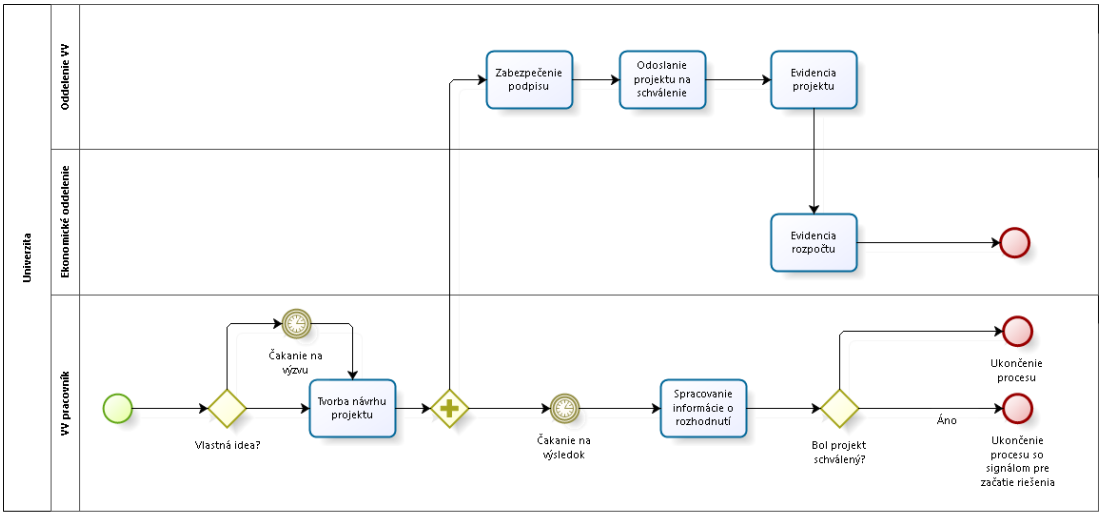
Za spomenutie stojí takisto modelovacia technika (jazyk) ArchiMate pre popis enterprise architektúry pomocou súboru konceptov a vzťahov medzi doménami architektúry. Podľa [51] ArchiMate v porovnaní s MDA popisuje enterprise architektúru z hľadiska vyššej úrovne abstrakcie a používa menej detailov. Takisto ArchiMate používa sémantický jazyk pre modelovanie enterprise architektúry namiesto jazyka UML, ktorý je použitý na syntaktickú reprezentáciu úrovne PIM a PSM v MDA.

Úplne iný prístup predstavuje nástroj jBPM pre manažment biznis procesov, ktorý umožňuje graficky modelovať biznis vrstvu. Takto vytvorené biznis procesy sú ihneď spustiteľné a pripravené na priamu implementáciu funkcionality a integráciu do IS bez nutnosti dodatočnej transformácie. Vo všeobecnosti je tento prístup vhodný na rýchlu implementáciu biznis logiky.

5.4.2 Modelovanie CIM a PIM úrovne pomocou ontológie

Rovina CIM býva často vyjadrená formou BPMN diagramov. Naproti tomu rovina PIM býva vyjadrená v jazyku UML. Obidve notácie (BPMN aj UML) sú značne rozsiahle a ich komplexné zachytenie by bolo nad rámec tejto práce. Preto sa autor zameria na niektoré vybrané a často používané elementy a diagramy.

Ako dáta pre výber elementov a tvorbu modelov budú slúžiť vybrané časti z procesu riadenia vedy a výskumu na Žilinskej univerzite.



Obrázok 17 – BPMN proces vybraných častí podania a schválenia vedecko-výskumného projektu na Žilinskej univerzite - spracované podľa projektu [52]

Vybrané BPMN elementy:

- Plavecká dráha,
- Bazén,

- Aktivita,
- Začiatok procesu,
- Koniec procesu,
- Rozhodovací blok,
- Paralelné vetvenie,
- Časovač.

Vybrané elementy UML diagramu prípadov použitia:

- Aktér,
- Prípad použitia,
- Asociácia.

Vzhľadom na uvedené elementy by model v rámci ontológie mal pozostávať z elementov, ktoré budú obsahovať nasledujúce prvky:

- ID elementu
- Názov
- Typ elementu
    - Začiatok (Start),
    - Koniec (End),
    - Aktivita (Activity),
    - Časovač (Timer),
    - Paralelný blok (Parallel),
    - Rozhodovací blok (Decision)

- ID Zdroja
- ID Cieľa
- Vykonávateľ
- Vlastník

5.4.3 Dáta pre ontologický model

Vzorové dáta pre naplnenie ontologického modelu sú vytvorené priamo z informácii pochádzajúcich z procesov riadenia vedy a výskumu na Žilinskej univerzite. Tieto procesy boli navrhnuté v rámci projektu *Rozvoj ľudských zdrojov s podporou integrovaného informačného systému na hodnotenie vedecko-výskumných výsledkov* [52]. Navrhnuté dáta zobrazuje Tabuľka 1.

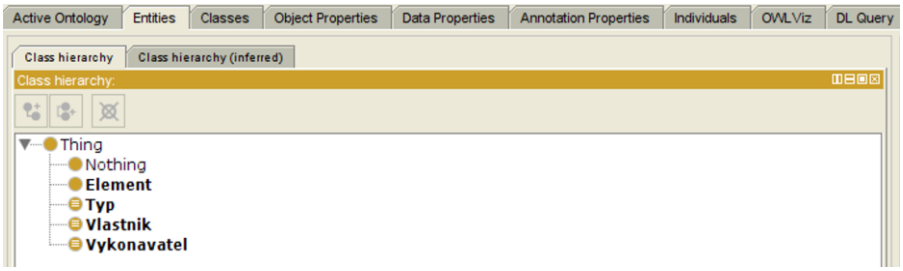
Tabuľka 1 – Návrh dát pre ontologický model

ID elementu:	1	**ID elementu:**	6	**ID elementu:**	11
Názov:	-	**Názov:**	Čakanie na výsledok	**Názov:**	Zabezpečenie podpisu
Typ:	Start	**Typ:**	Timer	**Typ:**	Activity
ID zdroja:	-	**ID zdroja:**	5	**ID zdroja:**	5
ID cieľa:	2	**ID cieľa:**	7	**ID cieľa:**	12
Vykonávateľ:	VV pracovník	**Vykonávateľ:**	VV pracovník	**Vykonávateľ:**	Oddelenie VV
Vlastník:	Univerzita	**Vlastník:**	Univerzita	**Vlastník:**	Univerzita
ID elementu:	2	**ID elementu:**	7	**ID elementu:**	12
Názov:	Vlastná idea?	**Názov:**	Spracovanie informácie o rozhodnutí	**Názov:**	Odoslanie projektu na schválenie
Typ:	Decision	**Typ:**	Activity	**Typ:**	Activity
ID zdroja:	1	**ID zdroja:**	6	**ID zdroja:**	11
ID cieľa:	3, 4	**ID cieľa:**	8	**ID cieľa:**	13
Vykonávateľ:	VV pracovník	**Vykonávateľ:**	VV pracovník	**Vykonávateľ:**	Oddelenie VV
Vlastník:	Univerzita	**Vlastník:**	Univerzita	**Vlastník:**	Univerzita
ID elementu:	3	**ID elementu:**	8	**ID elementu:**	13
Názov:	Čakanie na výzvu	**Názov:**	Bol projekt schválený	**Názov:**	Evidencia projektu
Typ:	Timer	**Typ:**	Decision	**Typ:**	Activity
ID zdroja:	2	**ID zdroja:**	7	**ID zdroja:**	12
ID cieľa:	4	**ID cieľa:**	9, 10	**ID cieľa:**	14
Vykonávateľ:	VV pracovník	**Vykonávateľ:**	VV pracovník	**Vykonávateľ:**	Oddelenie VV
Vlastník:	Univerzita	**Vlastník:**	Univerzita	**Vlastník:**	Univerzita
ID elementu:	4	**ID elementu:**	9	**ID elementu:**	14
Názov:	Tvorba návrhu projektu	**Názov:**	Ukončenie procesu	**Názov:**	Evidencia rozpočtu
Typ:	Activity	**Typ:**	End	**Typ:**	Activity
ID zdroja:	2, 3	**ID zdroja:**	8	**ID zdroja:**	13
ID cieľa:	5	**ID cieľa:**	-	**ID cieľa:**	15
Vykonávateľ:	VV pracovník	**Vykonávateľ:**	VV pracovník	**Vykonávateľ:**	Ekonomické oddelenie
Vlastník:	Univerzita	**Vlastník:**	Univerzita	**Vlastník:**	Univerzita
ID elementu:	5	**ID elementu:**	10	**ID elementu:**	15
Názov:	-	**Názov:**	Ukončenie procesu so signálom pre začatie riešenia	**Názov:**	-
Typ:	Parallel	**Typ:**	End(Y)	**Typ:**	End
ID zdroja:	4	**ID zdroja:**	8	**ID zdroja:**	14
ID cieľa:	6, 11	**ID cieľa:**	-	**ID cieľa:**	-
Vykonávateľ:	VV pracovník	**Vykonávateľ:**	VV pracovník	**Vykonávateľ:**	Ekonomické oddelenie
Vlastník:	Univerzita	**Vlastník:**	Univerzita	**Vlastník:**	Univerzita

5.4.4 Ontologický model

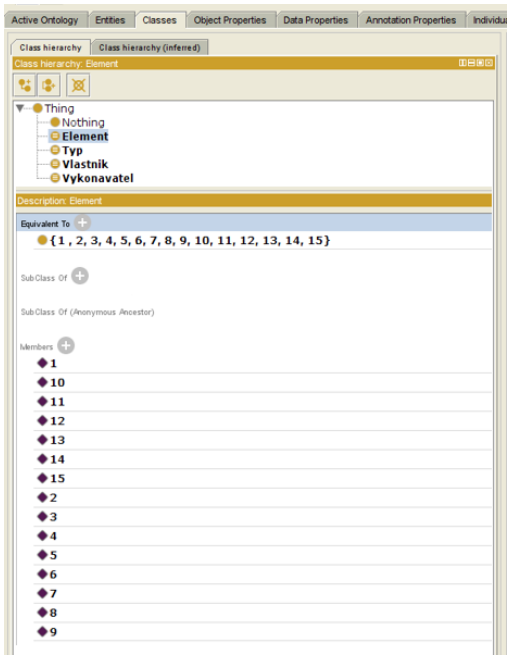
Na základe návrhu dát (Tabuľka 1) bol následne v nástroji Protege 4.3 (ontologický editor) vytvorený konkrétny ontologický model zapísaný v jazyku OWL. Tvorba modelu pozostáva z nasledovných krokov:

1. V rámci Entít (Entities) sú definované základné objekty, ktoré sa budú v modeli používať – Element, Typ, Vlastník a Vykonávateľ



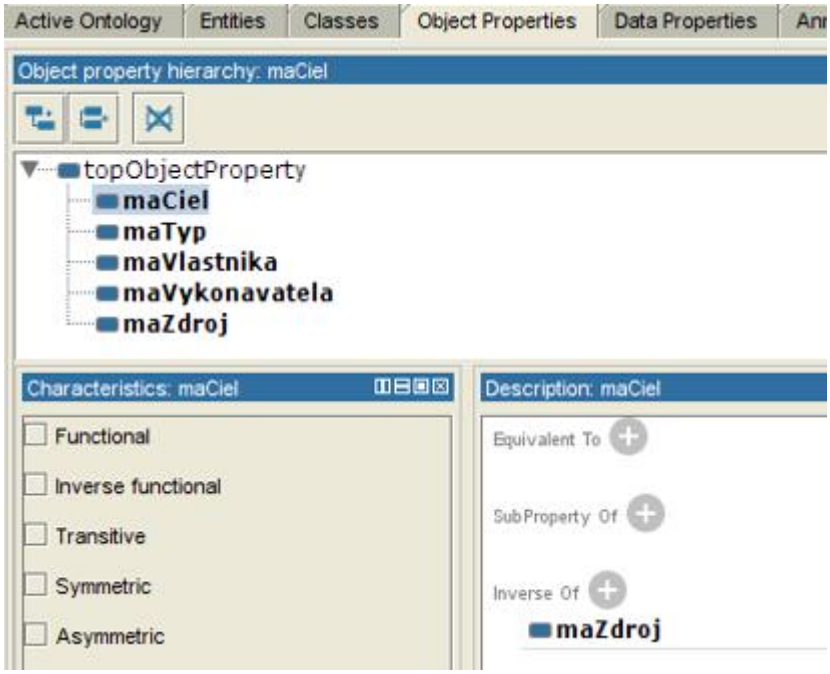
Obrázok 18 - Entity

2. Triedy (Classes) definujú vlastnosti entít – napríklad ID elementov 1-15



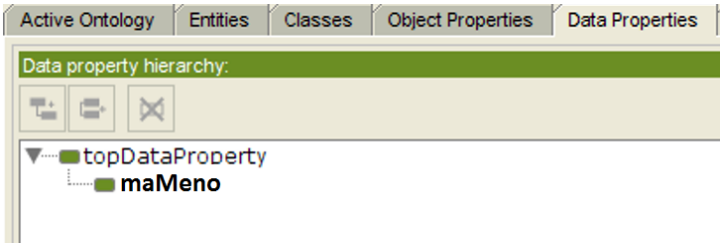
Obrázok 19 – Triedy

3. Vlastnosti objektov (Object Properties) definujú vlastnosti, cez ktoré sú objekty previazané. Na základe navrhnutých dát sú to vlastnosti *maCiel, maTyp, maVlastnika, maVykonavatela, maZdroj*



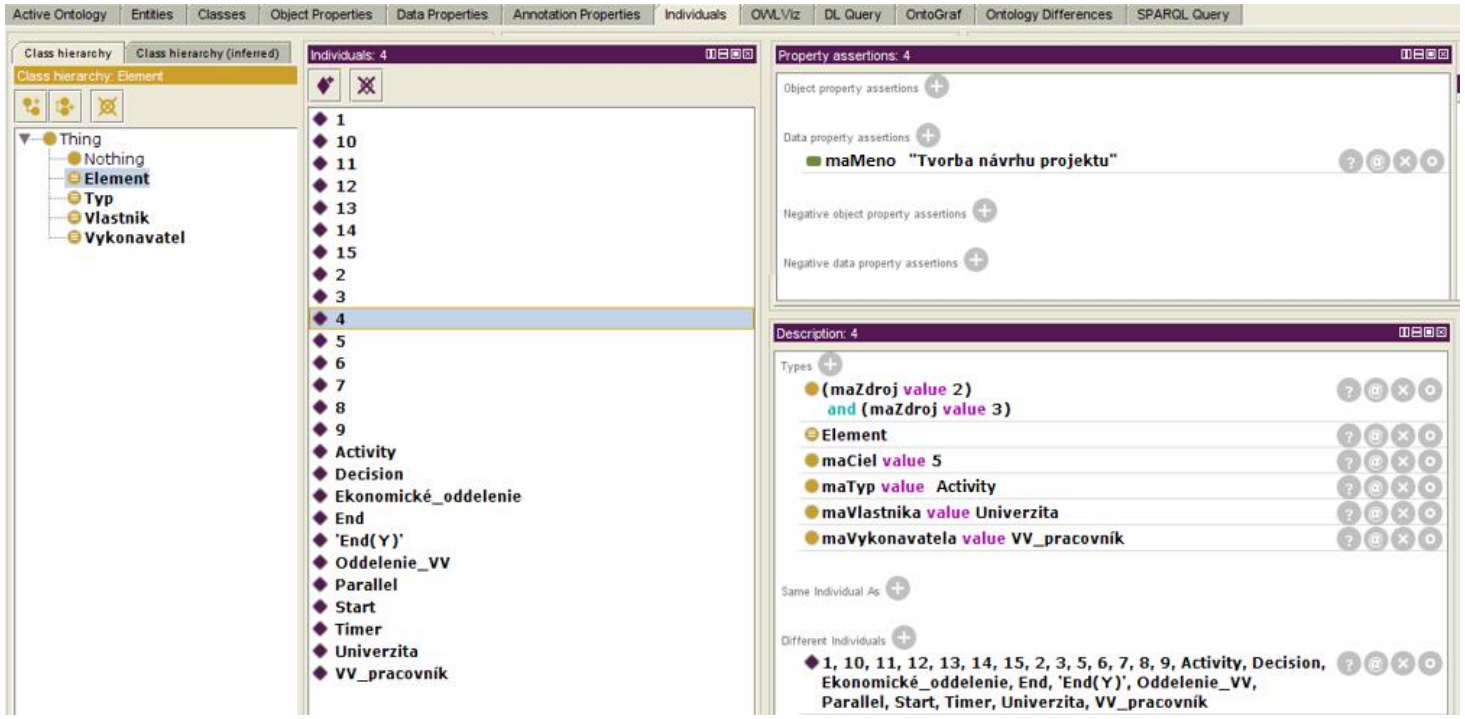
Obrázok 20 - Vlastnosti objektov

4. Pre Dátové vlastnosti (Data Properties) je definovaná jedna vlastnosť – *maMeno*, v rámci ktorej búdu ukladané názvy elementov.



Obrázok 21 - Dátové vlastnosti

5. Posledným krokom je vytvorenie Individualít (Individuals). Každý element má priradený zdroj(e), cieľ, vlastníka a vykonávateľa a meno (ak existuje). Ako znázorňuje Obrázok 22, element s ID 4 - *Tvorba návrhu projektu* obsahuje všetky dáta definované v návrhu (Tabuľka 1). To isté platí aj pre ostatné elementy.



Obrázok 22 - Individuality

Keďže vytvorený OWL súbor je príliš rozsiahly na to, aby sa dal prehľadne zobraziť v tejto práci, na ukážku bola vybraná časť zápisu, ktorá sa venuje spomínanému elementu s ID 4. Relevantné časti sú zvýraznené žltou farbou.

```
<!-- http://.../ontologies/2016/3/UniverzitaOntology#4 -->
   <owl:NamedIndividual
rdf:about="http://.../ontologies/2016/3/UniverzitaOntology#4">
      <rdf:type
rdf:resource="http://.../ontologies/2016/3/UniverzitaOntology#Element" />
      <rdf:type>
         <owl:Restriction>
            <owl:onProperty
rdf:resource="http://.../ontologies/2016/3/UniverzitaOntology#maTyp" />
            <owl:hasValue
rdf:resource="http://.../ontologies/2016/3/UniverzitaOntology#Activity" />
         </owl:Restriction>
      </rdf:type>
      <rdf:type>
         <owl:Restriction>
            <owl:onProperty
rdf:resource="http://.../ontologies/2016/3/UniverzitaOntology#maVlastnika" />
            <owl:hasValue
rdf:resource="http://.../ontologies/2016/3/UniverzitaOntology#Univerzita" />
         </owl:Restriction>
      </rdf:type>
      <rdf:type>
         <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
               <owl:Restriction>
                  <owl:onProperty
rdf:resource="http://.../ontologies/2016/3/UniverzitaOntology#maZdroj" />
                  <owl:hasValue
rdf:resource="http://.../ontologies/2016/3/UniverzitaOntology#2" />
               </owl:Restriction>
               <owl:Restriction>
                  <owl:onProperty
rdf:resource="http://.../ontologies/2016/3/UniverzitaOntology#maZdroj" />
                  <owl:hasValue
rdf:resource="http://.../ontologies/2016/3/UniverzitaOntology#3" />
               </owl:Restriction>
            </owl:intersectionOf>
         </owl:Class>
      </rdf:type>
      <rdf:type>
         <owl:Restriction>
            <owl:onProperty
rdf:resource="http://.../ontologies/2016/3/UniverzitaOntology#maCiel" />
            <owl:hasValue
rdf:resource="http://.../ontologies/2016/3/UniverzitaOntology#5" />
         </owl:Restriction>
      </rdf:type>
      <rdf:type>
         <owl:Restriction>
            <owl:onProperty
rdf:resource="http://.../ontologies/2016/3/UniverzitaOntology#maVykonavatela"
/>
            <owl:hasValue
rdf:resource="http://.../ontologies/2016/3/UniverzitaOntology#VV pracovník" />
         </owl:Restriction>
      </rdf:type>
      <maMeno>Tvorba návrhu projektu</maMeno>
   </owl:NamedIndividual>
```

5.4.5 Mapovanie ontológie na úroveň CIM

Ako už bolo spomínané v podkapitole 5.4.2, úroveň CIM býva reprezentovaná biznis modelmi. Aby mohla ontológia slúžiť pre potreby modelovania úrovne CIM je potrebné z ontologického modelu, ktorý je reprezentovaný súborom vo formáte OWL vytvoriť biznis model.

Na tvorbu týchto modelov je využívaná notácia BPMN (štandardizovaná skupinou OMG), ktorá poskytuje univerzálnu grafickú notáciu pre modelovanie biznis procesov. Na ukladanie BPMN modelov sa využíva štandard XPDL (štandardizovaný skupinou WfMC), ktorý umožňuje uložiť celú definíciu BPMN procesu formou špecifických XML súborov. XPDL predstavuje všeobecne akceptovaný formát, ktorý je využívaný v nástrojoch pre modelovanie biznis procesov. Tieto nástroje dokážu otvoriť XPDL súbory a zobraziť takto zapísané biznis procesy priamo v grafickej forme.

Medzi často používané XPDL prvky napríklad patria:

- <Package /> - obsahuje XMLNS deklaráciu
- <PackageHeader /> - popisuje napríklad verziu XPDL či názov dokumentu
- <Pools /> - bazény
- <Lanes /> - plavecké dráhy
- <StartEvent /> - začiatok procesu
- <EndEvent /> - ukončenie procesu
- <IntermediateEvent /> - prechodná udalosť
- <Task /> - aktivita
- <Route /> - vetvenie
- <Transition /> - prepojenia medzi jednotlivými elementami
- <NodeGraphicsInfos /> - obsahuje informácie pre vykreslenie elementov

Na to, aby bolo možné mapovať ontologický model na biznis procesy s elementami špecifikovanými v podkapitole 5.4.2 - *Plavecká dráha*, *Bazén*, *Aktivita, Začiatok procesu, Koniec procesu, Rozhodovací blok, Paralelné vetvenie* a *Časovač* je potrebné definovať mapovacie pravidlá a štruktúru pre XPDL súbor.

Mapovanie využíva *Vlastnosti objektov* (Object properties) definované v ontologickom modeli:

1. Plavecká dráha sa mapuje cez *maVykonavatela*
2. Bazén sa mapuje cez *maVlastnika*
3. Mapovanie typov elementov prebieha cez vlastnosť *maTyp*
    - Aktivita - hodnota *Activity*
    - Začiatok - hodnota *Start*
    - Koniec - hodnota *End*
    - Rozhodovací blok - hodnota *Decision*
    - Paralelné vetvenie - hodnota *Parallel*
    - Časovač - hodnota *Timer*
4. Názvy elementov sú mapované cez vlastnosť *maMeno*
5. Mapovanie prepojení elementov je riešené cez vlastnosti *maZdroj* a *maCiel*

Štruktúra XPDL musí v tomto prípade obsahovať najmä elementy ako: *Pools, Pool, Lanes, Lane, WorkflowProcesses, WorkflowProcess, Activity, Event, StartEvent, EndEvent, IntermediateIvent, Condition, Route, Transition.*

Na vytvorenie biznis modelu z ontologického modelu bola vytvorená aplikácia, ktorá z OWL súboru vygeneruje XPDL dokument popisujúci biznis proces. Aplikácia je vytvorená v jazyku Java a na prácu s OWL súborom využíva knižnicu OWL API. Na vytvorenie XPDL dokumentu sú následne použité mapovacie pravidlá uvedené vyššie.

Výslednú štruktúru vygenerovaného XPDL dokumentu z OWL súboru je možné vidieť nižšie.

```
<?xml version="1.0" encoding="utf-8"?>
<Package xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Id="a6bb45c8-cf57-43be-8107-
68adb4e3b16b" Name="Diagram 1" xmlns="http://www.wfmc.org/2009/XPDL2.2">
  <PackageHeader>
    <XPDLVersion>2.2</XPDLVersion>
    <Description>Diagram 1</Description>
  </PackageHeader>
  <ExternalPackages />
  <Pools>
    <Pool Id="pool1" Name="Univerzita" Process="proc1" BoundaryVisible="true">
      <Lanes>
        <Lane Id="lane1" Name="Oddelenie VV" ParentPool="pool1">
        </Lane>
        <Lane Id="lane2" Name="Ekonomické oddelenie" ParentPool="pool1">
        </Lane>
        <Lane Id="lane3" Name="VV pracovník" ParentPool="pool1">
        </Lane>
      </Lanes>
    </Pool>
  </Pools>
  <WorkflowProcesses>
    <WorkflowProcess Id="proc1" Name="Univerzita">
      <Activities>
        <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e001" Name="">
          <Event>
            <StartEvent Trigger="None" />
          </Event>
        </Activity>
        <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e014" Name="Evidencia rozpočtu">
        </Activity>
        <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e003" Name="Čakanie na výzvu">
           <Event>
            <IntermediateEvent Trigger="Timer">
            </IntermediateEvent>
          </Event>
        </Activity>
        <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e004" Name="Tvorba návrhu projektu">
        </Activity>
        <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e002" Name="Vlastná idea?">
          <Route />
        </Activity>
        <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e005" Name="">
          <Route GatewayType="Parallel" />
        </Activity>
        <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e006" Name="Čakanie na výsledok">
          <Event>
```

```
      <IntermediateEvent Trigger="Timer">
      </IntermediateEvent>
    </Event>
  </Activity>
  <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e007" Name="Spracovanie informácie o rozhodnutí">
  </Activity>
  <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e008" Name="Bol projekt schválený?">
    <Route />
  </Activity>
  <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e009" Name="Ukončenie procesu">
    <Event>
      <EndEvent Result="None" />
    </Event>
  </Activity>
  <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e010" Name="Ukončenie procesu so signálom pre začatie riešenia">
    <Event>
      <EndEvent Result="None" />
    </Event>
  </Activity>
  <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e011" Name="Zabezpečenie podpisu">
  </Activity>
  <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e012" Name="Odoslanie projektu na schválenie">
  </Activity>
  <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e013" Name="Evidencia projektu">
  </Activity>
  <Activity Id="f4630b3d-0d9b-48d6-bd55-e79fbc69e015" Name="">
    <Event>
      <EndEvent Result="None" />
    </Event>
  </Activity>
</Activities>
<Transitions>
  <Transition Id="tr1" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e001" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e002">
  </Transition>
  <Transition Id="tr2" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e002" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e003">
    <Condition Type="CONDITION">
    </Condition>
  </Transition>
  <Transition Id="tr3" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e003" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e004">
  </Transition>
  <Transition Id="tr4" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e002" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e004">
    <Condition Type="CONDITION">
    </Condition>
  </Transition>
  <Transition Id="tr5" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e004" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e005">
  </Transition>
```

```
      <Transition Id="tr6" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e005" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e006">
      </Transition>
      <Transition Id="tr7" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e006" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e007">
      </Transition>
      <Transition Id="tr8" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e007" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e008">
      </Transition>
      <Transition Id="tr9" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e008" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e009">
        <Condition Type="CONDITION">
        </Condition>
      </Transition>
      <Transition Id="tr10" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e008" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e010" Name="Áno">
        <Condition Type="CONDITION">
        </Condition>
      </Transition>
      <Transition Id="tr11" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e005" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e011">
      </Transition>
      <Transition Id="tr12" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e011" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e012">
      </Transition>
      <Transition Id="tr13" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e012" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e013">
      </Transition>
      <Transition Id="tr14" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e013" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e014">
      </Transition>
      <Transition Id="tr15" From="f4630b3d-0d9b-48d6-bd55-e79fbc69e014" To="f4630b3d-0d9b-48d6-bd55-e79fbc69e015">
      </Transition>
    </Transitions>
  </WorkflowProcess>
</WorkflowProcesses>
     </Package>
```

Tento XPDL súbor je možné otvoriť priamo v modelovacom nástroji Bizagi Modeler a tak zobraziť biznis proces v plne grafickej podobe. V rámci generovania XPDL súboru nebol riešený prepočet rozmiestnenia jednotlivých elementov. Správne grafické rozmiestnenie rôzne previazaných elementov nie je triviálny problém, a jeho riešenie je nad rámec cieľov tejto práce. Tento fakt však nijako zásadne neobmedzuje prehľadnosť načítaného XPDL modelu, pretože používateľ si vie veľmi ľahko a rýchlo sám rozmiestniť BPMN elementy podľa svojich preferencii. Navyše, po následnom uložení súboru ostane definované rozmiestenie uložené priamo v XPDL súbore, takže tento úkon stačí vykonať iba raz.

5.4.6 Mapovanie ontológie na úroveň PIM

Jedným z modelov, ktorý sa používa na reprezentovanie úrovne platformovo nezávislého modelu – PIM je aj diagram prípadov použitia. Diagramy vyjadrené v modelovacom jazyku UML (medzi ktoré patrí aj diagram prípadov použitia) bývajú ukladané vo formáte XMI – jedná sa o všeobecne uznávaný štandard vydaný skupinou OMG.

Na to, aby bolo možné mapovať ontologický model do diagramu prípadov použitia s elementami špecifikovanými v podkapitole 5.4.2 - *Aktér, Prípad použitia, Asociácia* je potrebné definovať mapovacie pravidlá a štruktúru pre XMI súbor.

Mapovanie využíva *Vlastnosti objektov* (Object properties) definované v ontologickom modeli:

1. Aktéri sa mapujú cez *maVykonavatela*
2. Prípady použitia sa mapujú cez *maTyp* – hodnota *Activity*
3. Mapovanie asociácii vychádza z predchádzajúcich dvoch bodov, pretože každá aktivita má priradeného vykonávateľa - tým pádom je možné jednoznačne určiť asociáciu.

Štruktúra XMI súborov dokáže byť pomerne zložitá a môže obsahovať veľké množstvo elementov a atribútov. Hľadanie vhodnej XMI štruktúry prebiehalo na základe experimentov. Okrem všeobecných atribútov (názov, verzia, typ diagramu, ...) je pre zapísanie elementov *Aktér, Prípad použitia, Asociácia* potrebné použiť atribúty *uml:Actor, uml:UseCase, uml:Association* a *uml:Model* pre vytvorenie hraníc systému.

Na základe uvedených poznatkov bola aplikácia popísaná v podkapitole 5.4.5 upravená a rozšírená o ďalšie mapovacie pravidlá aby bolo možné generovať XMI súbor.

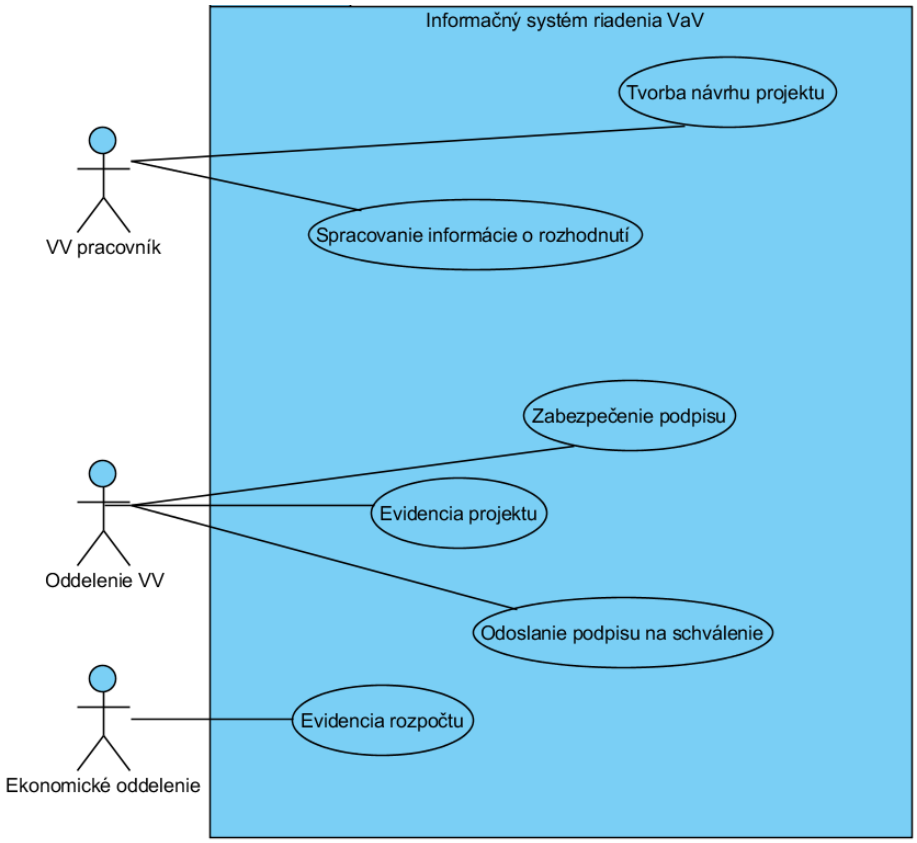
Takto vygenerovaný XMI súbor vyzerá nasledovne:

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmi:version="2.1">
   <uml:Model xmlns:uml="http://schema.omg.org/spec/UML/2.0" name="Podanie a schválenie projektu" xmi:id="f4630b3d-0d9b-48d6-bd55-
e79fbc69e000" decription="Podanie a schválenie projektu">
      <ownedMember name="VV pracovník" visibilty="public" xmi:id="actor1" xmi:type="uml:Actor" />
      <ownedMember name="Ekonomické oddelenie" visibilty="public" xmi:id="actor2" xmi:type="uml:Actor" />
      <ownedMember name="Oddelenie VV" visibilty="public" xmi:id="actor3" xmi:type="uml:Actor" />
      <ownedMember name="Tvorba návrhu projektu" xmi:id="f4630b3d-0d9b-48d6-bd55-e79fbc69e001" xmi:type="uml:UseCase" />
      <ownedMember name="Spracovanie informácie o rozhodnutí" xmi:id="f4630b3d-0d9b-48d6-bd55-e79fbc69e002" xmi:type="uml:UseCase" />
         <ownedMember name="Evidencia rozpočtu" xmi:id="f4630b3d-0d9b-48d6-bd55-e79fbc69e003" xmi:type="uml:UseCase" />
      <ownedMember name="Zabezpečenie podpisu" xmi:id="f4630b3d-0d9b-48d6-bd55-e79fbc69e004" xmi:type="uml:UseCase" />
      <ownedMember name="Odoslanie podpisu na schválenie" xmi:id="f4630b3d-0d9b-48d6-bd55-e79fbc69e005" xmi:type="uml:UseCase" />
      <ownedMember name="Evidencia projektu" xmi:id="f4630b3d-0d9b-48d6-bd55-e79fbc69e006" xmi:type="uml:UseCase" />

      <ownedMember name="Informačný Systém riadenia VaV" xmi:id="js0FuCqFYGTeyQis" xmi:type="uml:Model">
          <xmi:Extension extender="Visual Paradigm">
              <system/>
              <isRoot xmi:value="false"/>
              <qualityScore value="-1"/>
          </xmi:Extension>
      </ownedMember>
      <ownedMember xmi:id="EO1AE_EA301" xmi:type="uml:Association">
         <memberEnd xmi:idref="StarTAsoC_EA301" />
         <ownedEnd association="EO1AE_EA301"  type="actor1" xmi:id="StarTAsoC_EA301" xmi:type="uml:Property" />
         <memberEnd xmi:idref="EnDAsoC_EA301" />
         <ownedEnd association="EO1AE_EA301"  type="f4630b3d-0d9b-48d6-bd55-e79fbc69e001" xmi:id="EnDAsoC_EA301" xmi:type="uml:Property" />
      </ownedMember>
      <ownedMember xmi:id="EO1AE_EA302" xmi:type="uml:Association">
         <memberEnd xmi:idref="StarTAsoC_EA302" />
         <ownedEnd association="EO1AE_EA302"  type="actor1" xmi:id="StarTAsoC_EA302" xmi:type="uml:Property" />
         <memberEnd xmi:idref="EnDAsoC_EA302" />
         <ownedEnd association="EO1AE_EA302"  type="f4630b3d-0d9b-48d6-bd55-e79fbc69e002" xmi:id="EnDAsoC_EA302" xmi:type="uml:Property" />
      </ownedMember>
      <ownedMember xmi:id="EO1AE_EA303" xmi:type="uml:Association">
         <memberEnd xmi:idref="StarTAsoC_EA303" />
         <ownedEnd association="EO1AE_EA303"  type="actor3" xmi:id="StarTAsoC_EA303" xmi:type="uml:Property" />
         <memberEnd xmi:idref="EnDAsoC_EA303" />
         <ownedEnd association="EO1AE_EA303"  type="f4630b3d-0d9b-48d6-bd55-e79fbc69e004" xmi:id="EnDAsoC_EA303" xmi:type="uml:Property" />
      </ownedMember>
      <ownedMember xmi:id="EO1AE_EA304" xmi:type="uml:Association">
         <memberEnd xmi:idref="StarTAsoC_EA304" />
         <ownedEnd association="EO1AE_EA304"  type="actor3" xmi:id="StarTAsoC_EA304" xmi:type="uml:Property" />
         <memberEnd xmi:idref="EnDAsoC_EA304" />
         <ownedEnd association="EO1AE_EA304"  type="f4630b3d-0d9b-48d6-bd55-e79fbc69e005" xmi:id="EnDAsoC_EA304" xmi:type="uml:Property" />
```

```
      </ownedMember>
      <ownedMember xmi:id="EO1AE_EA305" xmi:type="uml:Association">
         <memberEnd xmi:idref="StarTAsoC_EA305" />
         <ownedEnd association="EO1AE_EA305"  type="actor3" xmi:id="StarTAsoC_EA305" xmi:type="uml:Property" />
         <memberEnd xmi:idref="EnDAsoC_EA305" />
         <ownedEnd association="EO1AE_EA305"  type="f4630b3d-0d9b-48d6-bd55-e79fbc69e006" xmi:id="EnDAsoC_EA305" xmi:type="uml:Property" />
      </ownedMember>
      <ownedMember xmi:id="EO1AE_EA306" xmi:type="uml:Association">
         <memberEnd xmi:idref="StarTAsoC_EA306" />
         <ownedEnd association="EO1AE_EA306"  type="actor2" xmi:id="StarTAsoC_EA306" xmi:type="uml:Property" />
         <memberEnd xmi:idref="EnDAsoC_EA306" />
         <ownedEnd association="EO1AE_EA306"  type="f4630b3d-0d9b-48d6-bd55-e79fbc69e003" xmi:id="EnDAsoC_EA306" xmi:type="uml:Property" />
      </ownedMember>
   </uml:Model>
   <uml:Diagram xmlns:uml="http://schema.omg.org/spec/UML/2.0" diagramType="UseCaseDiagram" documentation="" name="Podanie a schválenie
projektu" toolName="" xmi:id="f4630b3d-0d9b-48d6-bd55-e79fbc69e" />
       </xmi:XMI>
```

Následne je možné tento súbor otvoriť v modelovacom nástroji, ktorý podporuje UML notáciu a umožňuje import vo formáte XMI. Grafickú formu diagramu prípadov použitia po importovaní XMI súboru do nástroja Visual Paradigm znázorňuje Obrázok 23 uvedený nižšie.



Obrázok 23 - Podanie a schválenie projektu - diagram prípadov použitia

5.4.7 Zhrnutie

Prezentované riešenie mapovania ontológie na úroveň CIM a PIM ukázalo, ako je možné využiť ontológiu pre jednotné modelovanie spomínaných úrovní v rámci MDA.

Ontologický model je možné mapovať na *Model nezávislý na počítačovom spracovaní* (CIM) a aj na *Model nezávislý na platforme* (PIM). Následne je možné tieto modely zobraziť v plnohodnotnej grafickej forme – či už sa jedná o biznis procesy v notácii BPMN alebo prípady použitia v notácii UML. Týmto spôsobom je možné zrýchliť modelovací proces a takisto odstrániť potrebu transformácie medzi úrovňami CIM a PIM. Na to, aby bolo možné odstrániť potrebu transformácie v plnom rozsahu by bolo nutné dané mapovanie rozšíriť na celé spektrum problematiky vytvárania modelov v úrovniach CIM a PIM s ohľadom na všetky notácie, obmedzenia, konvencie, typy

súborov a štandardy. Napriek tomu súčasné riešenie dokáže zjednotiť, zrýchliť a sprehľadniť návrh, tvorbu a údržbu biznis procesov a diagramov prípadov použitia z hľadiska CIM/PIM transformácii a poskytuje priestor pre ďalší rozvoj v tejto výskumnej oblasti.

Výhodou navrhnutého riešenia je aj to, že prípadné zmeny v špecifikácii počas návrhu IS stačí meniť iba v ontologickom modeli a následne sa zmeny potom cez mapovanie na CIM/PIM úroveň prejavia priamo vo výsledných (grafických) modeloch, s ktorými môžu následne pracovať napríklad biznis používatelia, systémoví architekti či priamo vývojári. Vďaka tomuto faktu nie je potrebné prácne manuálne upravovať všetky vytvorené modely pri každej zmene.

6 Zhodnotenie riešenia

Potreba pracovať súčasne s informáciami, dátami aj znalosťami vytvára nové výzvy, vyriešenie ktorých môže posúvať vývoj informačných systémov na vyššiu úroveň.

Použitie ontológii pri vývoji IS je úzko späté s ontologickým a znalostným inžinierstvom. Ontologické inžinierstvo sa v niektorých ohľadoch veľmi nelíši od procesu vývoja informačných systémov, ktorý je charakteristický analýzou, tvorbou systémových modelov, a tvorbou logického a fyzického návrhu systému. Návrh previazania modelom riadenej architektúry s ontológiu je prvým prínosom tejto práce.

S neustále pribúdajúcimi dátami, informáciami a znalosťami sa dopyt po znalostných systémoch a manažmente znalostí zvyšuje. Napriek tomu nebola všeobecnému postupu návrhu znalostných systémov doteraz venovaná väčšia pozornosť. Vytvorenie architektonického rámca podľa princípov normy ISO/IEC/IEEE 42010:2011 *Systems and Software Engineering* pre vývoj znalostných systémov je druhým prínosom. Následné aplikovanie vytvoreného architektonického rámca pre špecifickú doménu s experimentálnym overením analýzy a návrhu IS je tretím prínosom tejto práce.

Štvrtým prínosom tejto dizertačnej práce je pozitívne overenie predpokladu, že ontológia by mohla byť použitá ako modelovací jazyk pre roviny CIM (Computer Independent Model) a PIM (Platform Independent Model) v modelom riadenej architektúre.

Uvedené prínosy otvárajú nové možnosti pre vývoj informačných systémov z hľadiska ich všeobecného návrhu (s dôrazom na znalostné systémy) a z hľadiska modelom riadenej architektúry a transformácii rovín CIM a PIM.

Záver

Neustály rozvoj IKT dáva možnosť vzniku rôznych typov informačných systémov, ktoré podporujú požiadavky používateľov všetkých domén. Vytvoriť informačný systém, ktorý by presne spĺňal požiadavky používateľov, je často neľahká úloha.

Vývoj informačných systémov je v súčasnosti založený na modeloch. Pojem model označuje vyjadrenie, podľa ktorého je spracovávaná každá etapa vývoja. Princíp modelom riadenej architektúry tento vývoj rozdeľuje do štyroch etáp/rovín. Každá rovina má svoje modelovacie jazyky a medzi jednotlivými rovinami sú vytvárané transformácie, ktoré postupne transformujú požiadavky z najvyššej roviny CIM – *Computer Independent Model* cez PIM - *Platform Independent Model*, PSM - *Platform Specific Model* až do modelu zdrojového kódu. Používané modely sú prevažne grafické a pre komunikáciu medzi nimi sú vytvárané rôzne typy transformácií. Ontológia ako disciplína je využívaná aj pri vývoji rôznych typov informačných systémov.

Cieľom tejto dizertačnej práce bolo navrhnúť architektonický rámec riešenia informačného systému pre riadenie znalostí v organizácii použitím princípov modelom riadenej architektúry (MDA) a ontológií ako modelovacieho jazyka v rovinách CIM (Computer Independent Model) a PIM (Platform Independent Model). Dosiahnutie uvedeného cieľa bolo rozčlenené do štyroch parciálnych cieľov:

1. Začleniť princípy MDA do postupov ontologického a znalostného inžinierstva.
2. Vytvoriť architektonický rámec pre vývoj znalostných systémov s použitím ontológií s dôrazom na modelovanie roviny CIM a PIM.
3. Aplikovať vytvorený architektonický rámec pre špecifickú doménu s experimentálnym overením analýzy a návrhu IS.
4. Zdôvodniť a potvrdiť využitie ontológií v modelom riadenej architektúre oproti iným modelovacím jazykom je štvrtý parciálny cieľ.

Ciele sú vypracované v rámci riešenia práce v kapitole 5. Po začlenení princípov modelom riadenej architektúry do znalostného inžinierstva bol vytvorený architektonický rámec pre vývoj znalostných systémov, ktorý zohľadňuje modelovanie roviny CIM a PIM. Výsledným riešením experimentálneho overenia je systém pre podporu plánovania procesov priemyselnej výroby. V rámci 4. parciálneho cieľa bola navrhnutá dátová štruktúra a vytvorený ontologický model. Konkrétne dáta boli prevzaté z procesu riadenia vedy a výskumu na Žilinskej univerzite. Ontologický model, ktorého účelom bolo navzájom integrovať informácie z rovín CIM a PIM bol vytvorený v jazyku OWL. Po definovaní transformačných pravidiel a súborových štruktúr bol model mapovaný na biznis procesy v jazyku BPMN a prípady použitia v jazyku UML. Takto vytvorené modely je možné načítať v modelovacích nástrojoch a zobraziť ich priamo v grafickej podobe. Toto riešenie podporuje predpoklad vyslovený v kapitole 2 – *Ontológia by mohla byť vhodnejším nástrojom pre uplatnenie princípov v MDA, pretože nebude potrebné vytvárať transformačné vzťahy medzi CIM a PIM rovinou*. Bežne používané modelovacie nástroje síce umožňujú tvorbu modelov do najmenších detailov, avšak neriešia problematiku priamej transformácie medzi vytvorenými modelmi.

Použite ontologického modelu oproti iným spôsobom uloženia je odôvodené tým, že ontológia nie je iba úložiskom ale poskytuje mechanizmy pre reprezentáciu dát, informácii a znalostí, ich vzájomné prepájanie a vyvodzovanie súvislostí. Vzhľadom na tento fakt možno v budúcnosti vytvorený ontologický model efektívne rozšíriť nielen o podporu iných modelovacích jazykov a notácii ale aj o aspekty, ktorých vzájomná integrácia v rámci návrhu a modelovania IS nebola doteraz uvažovaná. Môže to byť priama integrácia na projektový manažment, dokumentácie, existujúce dáta či dokonca celé informačné systémy. Toto je však iba niekoľko námetov na ďalšie možnosti smerovania v tejto oblasti.

Takisto netreba zabúdať na spomínanú problematiku týkajúcu sa grafického rozmiestnenia prvkov po načítaní vygenerovaných modelov, či na to, že samotná tvorba komplexných ontologických modelov vyžaduje rozsiahle skúsenosti znalostného inžiniera.

Použitá literatúra a zdroje

[1] J. A. HOFFER, J. F. GEORGE a J. S. VALACICH, Modern Systems Analysis and Design. 4th edition, Pearson, 2004.

[2] J. Habr a J. Vepřek, Systémová analýza a syntéza, Praha: SNTL, 1972.

[3] J. F. a. k. GEORGE, Object – Oriented Systems Analysis and Design. 2nd edition, Pearson, 2007.

[4] G. Firesmith, Capell, Falkenthal, B. Hammons, I. Latimer T a Merendino, The Method Framework for Engineering System Architectures, CRC Press, 2009.

[5] I. 42010:2011, *Systems and software engineering - Architecture description,* 2011.

[6] „Model-Based Engineering Forum," [Online]. Available: http://www.modelbasedengineering.com/.

[7] Modeling Languages, [Online]. Available: http://modeling-languages.com/clarifying-concepts-mbe-vs-mde-vs-mdd-vs-mda/.

[8] *Model-Based Engineering Forum.*

[9] Object Managemnet Group, [Online]. Available: http://www.omg.org/mda/specs.htm.

[10] MILLER a MUKERJI, „www.omg.org," 2011. [Online]. Available: http://www.omg.org/docs/omg/03-06-01.pdf.

[11] MILLER a FRANK, „MDA Guide revision Draft-Version 0.1.2," 2006.

[12] M. Kardoš a M. Drozdová, *Tranformácia CIM do PIM v modelom riadenej architektúre (MDA),* 2011.

[13] S. Kherraf, W. Suryn a É. Lefebvre , „Transformation from CIM to PIM Using Patterns and Archetypes," rev. *19th Australian Conference on Software Engineering*, 2008.

[14] A. Rodríguez, E. Fernández-Medina a M. Piattini, „CIM to PIM Transformation: A Reality," 2008.

[15] W. Zhang, H. Mei, H. Zhao a J. Yang, „Transformation from CIM to PIM: A Feature-Oriented Component-Based Approach," rev. *Model Driven Engineering Languages and Systems*, Springer, 2005, pp. 248-263.

[16] X.-X. Cao, H.-K. Miao a Q.-G. Xu, „Modeling and Refining the Service-Oriented Requirement,“ rev. *Sixth International Symposium on Theoretical Aspects of Software Engineering*, 2008.

[17] M. Kardoš, *Tranformácia CIM do PIM v modelom riadenej architektúre (MDA),* 2011.

[18] M. Kardoš a M. Drozdová, „Analytical method of CIM to PIM transformation in Model Driven Architecture (MDA),“ *Journal of Information and Organizational Sciences,* zv. 34, %1. vyd.1, pp. 89-99, Január 2010.

[19] B. Po, „3 reasons why the future of wearables is in software: Knowledge Grains,“ 28 Apríl 2014. [Online]. Available: http://www.ngrain.com/3-reasons-why-the-future-of-wearables-is-in-software/. [Cit. 9 Október 2014].

[20] A. Buchalcevová a L. Gála, „Architektura v podnikové informatice,“ *SYSTÉMOVÁ INTEGRACE,* 2008.

[21] Dongwoo, Jeongsoo, Sungchul a Kwangsoo, „An ontology-based Enterprise Architecture,“ *Expert Systems with Applications,* %1. vyd.37, 2010.

[22] Damjanović, „Semantic reengineering of business processes,“ *Information Systems,* %1. vyd.35, 2010.

[23] A. M. Hickey a A. M. Davis, „An Ontological Approach To Requirements Elicitation Technique Selection,“ rev. *ONTOLOGIES A Handbook of Principles, Concepts and Aplicationas in Information Systems*, New York, Springer, 2007.

[24] B. C. Stahl, „Ontology, Life-World, and Responsibility in IS,“ rev. *ONTOLOGIES A Handbook of Principles, Concepts and Aplicationas in Information Systems*, New York, Springer, 2007.

[25] M. Vargas-Vera, E. Moreale, A. Stutt, E. Motta a F. Ciravegna, „MNM: Semi-automatic ontology population from text,“ rev. *ONTOLOGIES A Handbook of Principles, Concepts and Aplicationas in Information Systems*, New York, Springer, 2007.

[26] D. N. Batanov a W. Vongdoiwang, „Use of Ontologies for Creating Object Oriented Models,“ rev. *ONTOLOGIES A Handbook of Principles, Concepts and Aplicationas in Information Systems*, New York, Springer, 2007.

[27] T. Fischer, „Conjunctive Query Programming: A Paradigm for Knowledge Engineering of Optimization Problems in the Semantic Web,“ rev. *KDO 2014, The 2014 IEEE/WIC/ACM International Conference on Web Intelligence - konferencia*, Varšava, 2014.

[28] M. Vajgl a J. Procházka, *Ontologické informační systémy a systémy na podporu rozhodování za neurčitosti.*

[29] D. Gašević, D. Djurić a V. Devedžić, Model Driven Architecture and Ontology Development, Berlin: Springer, 2006.

[30] R. Kishore a R. Sharman, „Computational Ontologies and Information Systems I: Foundations," *Communications of the Association for Information Systems,* 14 August 2004.

[31] R. Kishore, H. Zhang a R. Ramesh, *A Helix-Spindle model for ontological,* 2004.

[32] M. K. Bergman, *A Brief Survey of Ontology Development Methodologies,* 2010.

[33] F. Hussain, C. Lucas a M. A. Ali, „Managing Knowledge Effectively," *Journal of Knowledge Management Practice,* 2004.

[34] S. Denning, „What is knowledge? : SteveDenning.com," [Online]. Available: http://www.stevedenning.com/Knowledge-Management/what-is-knowledge.aspx. [Cit. 15 Máj 2014].

[35] C. W. Holsapple a A. B. Whinston, Decision Support Systems: A Knowledge Based Approach, West Publishing, 1996.

[36] M. H. Zack, „Developing A Knowledge Strategy," *California Management Review,* 1999.

[37] V. Varma, „Use of Ontologies for Organizational Knowledge Management and Knowledge Management Systems," rev. *ONTOLOGIES A Handbook of Principles, Concepts and Aplicationas in Information Systems*, New York, Springer, 2007.

[38] S. Kendal a M. Creen, An Introduction to Knowledge Engineering, London: Springer, 2007.

[39] B. Díaz-Agudo a P. A. Gonzáles-Calero, „An Ontological Approach To Develop Knowledge Intensive CBR Systems," rev. *ONTOLOGIES A Handbook of Principles, Concepts and Aplicationas in Information Systems*, New York, Springer, 2007.

[40] B. Engel, „Agricultural Systems Engineering: College of Engineering," [Online]. Available: https://engineering.purdue.edu/~engelb/abe565. [Cit. 1 Júl 2014].

[41] Good e-Learning Blog, „Common BPMN modeling mistakes: Swimlanes," [Online]. Available: http://blog.goodelearning.com/bpmn/common-bpmn-modeling-mistakes-swimlanes/. [Cit. 7 Október 2014].

[42] J. Mervart, Základy metodologie vědy, Praha: Svoboda, 1977.

[43] ISO/IEC/IEEE 42010, „Home: Systems and software engineering — Architecture description,“ [Online]. Available: http://www.iso-architecture.org/ieee-1471/index.html. [Cit. 30 Marec 2015].

[44] Roger Sessions ObjectWatch, Inc., „A Comparison of the Top Four Enterprise-Architecture Methodologies: Microsoft developer network,“ Máj 2007. [Online]. Available: https://msdn.microsoft.com/en-us/library/bb466232.aspx. [Cit. 4 Apríl 2015].

[45] N. Rozanski a E. Woods, Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives, New Jersey: Pearson Education, Inc., 2005.

[46] J. Garland a R. Anthony, Large-Scale Software Architecture, Chichester: Wiley, 2003.

[47] A. Razavizadeh, H. Verjus, S. Cimpan a S. Ducasse, „IEEE/IFIP WICSA/ECSA,“ rev. *Multiple Viewpoints Architecture Extraction*, 2009.

[48] L. Gála, A. Buchalcevová a J. Jandoš, „Podniková architektura,“ 2012.

[49] M. Vajsová, M. Zábovský, M. Chochlík a K. Matiaško, „Databázové Systémy: Základy databázových systémov,“ 2008.

[50] „Elektroinštalačný materiál - Toms Hardware,“ 2016. [Online]. Available: http://www.tomshardware.sk/p/11262/vidlica-380v-sez-ivn-3253-32a-ip44-na-kabel.

[51] R. P. Seguel, „Is there a relation among Archimate and MDA?,“ 18 Marec 2016. [Online]. Available: http://rseguel.bpm-latam.org/2009/12/is-there-relation-among-archimate-and.html.

[52] Žilinská univerzita, ITMS 26110230063, *Rozvoj ľudských zdrojov s podporou integrovaného informačného systému na hodnotenie vedecko-výskumných výsledkov.*

[53] Gruber, 1992. [Online]. Available: http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.

[54] J. M. J. MILLER, *MDA Guide Version 1.0.1. OMG Inc.,* 2003.

[55] V. M. Procházka J., *Ontologické informační systémy a systémy na podporu rozhodování za neurčitosti,* 2010.

[56] B. Gupta, L. Iyer a J. E. Aronson, „Knowledge Management: Practices And Challenges,“ *Industrial Management And Data Systems,* %1. vyd.100, 2000.

[57] Mathias, „Knowledge Management: Scientometrics,“ 10 November 2008. [Online]. Available: scientometrics.wordpress.com/2008/11/10/knowledge-acquisition-bottleneck/. [Cit. 1 Júl 2014].

[58] J. Přikryl, „HN Online: Riadenie: Hľadanie ciest, ako zabezpečiť, aby praví ľudia mali v pravej chvíli pravé znalosti,“ 18 Marec 2004. [Online]. Available: http://hn.hnonline.sk/znalosti-nie-su-vediet-co-ale-vediet-ako-116099.

[59] A. Ferkov, „Knowledge Management - Robime IT,“ 2 Apríl 2013. [Online]. Available: http://robime.it/riadenie-znalosti-zabudana-cast-podnikovej-architektury/. [Cit. 17 September 2014].

[60] G. Meszaros, „Software architecture in BNR,“ rev. *Proceedings of the First International Workshop on Architectures for Software Systems*, 1995.