FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO
BRATISLAVA

PODPORA ĽUDSKÝCH ÚLOH V BIZNIS PROCESOCH ZAPÍSANÝCH V JAZYKU BPEL

Diplomová práca

Ľuboš Dobrovoda **2008**

PODPORA ĽUDSKÝCH ÚLOH V BIZNIS PROCESOCH ZAPÍSANÝCH V JAZYKU BPEL

DIPLOMOVÁ PRÁCA

Ľuboš Dobrovoda

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA INFORMATIKY

Informatika

Vedúci / školiteľ diplomovej práce
Pavol Mederly, Mgr.

BRATISLAVA 2008

Čestne vyhlasujem, že som túto prácu vypracoval samostatne, len s použitím uvedenej literatúry.

V Bratislave dňa 5. mája 2008 _____

Ľuboš Dobrovoda

Ďakujem diplomovému vedúcemu Mgr. Pavlovi Mederlymu za vedenie diplomovej práce, konzultácie a cenné rady.

Abstrakt

Autor:	Ľuboš Dobrovoda
Názov práce:	Podpora ľudských úloh v biznis procesoch zapísaných v jazyku bpel
Škola:	Univerzita Komenského v Bratislave
Fakulta:	Fakulta matematiky, fyziky a informatiky
Katedra:	Katedra informatiky
Vedúci diplomovej práce:	Mgr. Pavol Mederly
Rok:	2008
Rozsah práce:	68 strán

V diplomovej práci sa zaoberáme jazykmi na popis a prácu s biznis procesmi. Táto oblasť je relatívne nová a dynamicky sa rozvíjajúca. Z tohto dôvodu v práci prinášame prehľad dnes najčastejšie používaných jazykov v tejto oblasti – BPEL, BPMN, XPDL a YAWL a porovnávame ich navzájom z rôznych hľadísk. Ťažiskom práce je skúmanie podpory jazyka BPEL pre úlohy, ktoré sú určené pre ľudí, a to s prihliadnutím na špecifické problémy, ktoré s ľudskými úlohami súvisia. V práci intenzívne využívame informácie z referenčnej akademickej práce vytvorenej skupinou workflow patterns initiative, ktorá obsahuje katalóg najčastejšie sa opakujúcich vzorov v biznis procesoch.

Súčasťou práce je aj návrh a čiastočná implementácia systému určeného na podporu práce s biznis procesmi zapísanými v jazyku BPEL, ktorý rozširuje jeho možnosti o podporu úloh určených pre ľudí.

Kľúčové slová:

Biznis proces, workflow, BPEL, workflow resource patterns, ľudské úlohy

Abstract (English)

In this master thesis we deal with languages for describing and work with business processes. This field of study is relatively new and therefore evolves quite dynamically. For this reason we try in our work to make an overview of the currently most widely used languages in this area – BPEL, BPMN, XPDL and YAWL and compare them from several viewpoints. The focus of the work lies in examining the support for tasks, which are intended for people from the perspective of workflow languages with consideration of specific problems, that are connected with people. In the thesis we use plenty of information from the scholar reference work of the group created by the workflow patterns initiative consisting of list of the most frequently used repetitive patterns in business processes.

Part of our work is also design and partial implementation of a system supporting work with workflows written in the BPEL language, that are extended to support human tasks.

Keywords:

Business process, workflow, BPEL, workflow resource patterns, human tasks

Abstrakt (auf Deutsch)

In dieser Diplomaarbeit beschäftigen wir uns mit Sprachen für Beschreibung und Arbeit mit Arbeitsabläufen. Dieses Fachgebiet ist relativ neu und entwickelt sich dynamisch. Aus diesem Grund bemühen wir uns in unserer Arbeit um einen Überblick der heutzutage am meisten benutzten Sprachen auf diesem Gebiet – BPEL, BPMN, XPDL und YAWL und vergleichen sie untereinander aus mehreren Perpektiven. Der Schwerpunkt der Arbeit liegt in Untersuchung der Unterstützung von Aufgaben, die für Menschen bestimmt sind, aus der Sicht der Arbeitsabläufesprachen mit Berücksichtigung spezifischer Problemen, die mit menschlichen Aufgaben zusammenhängen. In der Arbeit benutzen wir reichlich Informationen aus der akademishen Referenzarbeit, die von der Gruppe workflow patterns initiative gechaffen wurde und die aus einem Kalatogen der am häufigsten benutzten sich wiederholenden Mustern in Arbeitsabläufen besteht.

Der Bestandteil der Arbeit ist auch Entwurf und Teilimplementierung von einem System bestimmt für Unterstützung von Arbeit mit Arbeitsabläufen in der BPEL Sprache, das ihre Möglichkeiten um Unterstützung der menschlichen Aufgaben erweitert.

Schlüsselwörter:

Geschäftsprozess, Arbeitsablauf, BPEL, workflow resource patterns, menschliche Aufgaben

Predhovor

Predkladaná práca má nasledujúce dva ciele:

1. urobiť prehľad a porovnanie v súčasnosti používaných jazykov na popis biznis procesov a posúdiť, ktorý z nich by mohol poskytnúť najvhodnejší základ na implementáciu systému na podporu biznis procesov na Univerzite Komenského v Bratislave (ďalej len „UK“),
2. navrhnúť a čiastočne implementovať čo najvšeobecnejší systém podporujúci využívanie biznis procesov s podporou ľudí na UK.

Primárnou motiváciou pre vznik tejto práce bola potreba Univerzity Komenského čiastočne automatizovať procesy zahŕňajúce aplikácie a ľudí a takto ich urýchliť. Až neskôr – počas písania práce a nezávisle od nej bol na univerzite zakúpený nástroj Progress Sonic BPEL server, ktorý umožňuje definovať a vykonávať biznis procesy zapísané v jazyku BPEL. Toto bol jeden z hlavných dôvodov, prečo sa v práci zameriavame najmä na tento jazyk.

Prínos našej práce vidíme vo vytvorení podrobného prehľadu jednotlivých jazykov pre biznis procesy, v analýze podpory workflow resource patterns rozšírením BPEL4People, ako aj v návrhu systému rozširujúceho jazyk BPEL o podporu práce s ľuďmi.

Pri písaní práce sme kombinovali štúdium literatúry (špecifikácie jednotlivých skúmaných jazykov, resp. ich rozšírení, vedecké publikácie, webovú stránku iniciatívy workflow patterns [8] a ďalšie zdroje, prevažne internetové) s vlastnou analytickou, špecifikačnou, návrhovou a implementačnou činnosťou.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Na začiatok uvedieme príklad biznis procesu používaného na UK a aj na jeho základe vysvetlíme definície niekoľkých základných pojmov, ktoré budeme využívať v našej práci.

Nadobúdanie, opravy a zhodnocovanie výpočtovej techniky na pracoviskách Rektorátu UK sa riadi príkazom rektora Univerzity Komenského, č. 4 / 2004, (http://www.uniba.sk/fileadmin/user_upload/editors/subory/legislativa/prikaz_rektora_2004_4.pdf), v ktorom je definovaný príslušný biznis proces takto[1]:

1. Zamestnanec, ktorý zistí, že pre riadne a efektívne plnenie svojich pracovných úloh potrebuje novú výpočtovú techniku, svoju žiadosť stručne popíše v na tento účel určenom dokumente a tento postúpi svojmu priamemu nadriadenému.
2. Nadriadený sa vyjadrí k správnosti dokumentu a následne ho postúpi technickému správcovi Integrovaného informačného a komunikačného systému Rektorátu UK.
3. Technický správca posudzuje doručený dokument iba vtedy, ak sú v ňom v primeranom rozsahu uvedené skutočnosti podľa ods. 1 a obsahuje stanovisko podľa ods. 2, inak ho vráti predkladateľovi. Technický správca sa vyjadrí k opodstatnenosti žiadosti a súvisiacim technickým záležitostiam. Kladne posúdenú žiadosť postúpi technický správca kvestorovi, odmietnutú žiadosť vráti predkladateľovi. Ak technický správca nepodá svoje stanovisko k žiadosti do 14 dní, je táto postúpená jeho priamemu nadriadenému – riaditeľovi CIT UK, ktorý sa vyjadruje k žiadosti a predkladateľovi je o tom poslaná informácia.
4. Ak kvestor UK zistí súlad žiadosti s finančnými možnosťami UK, postúpi žiadosť na realizáciu.

Teraz vysvetlíme základné pojmy, s ktorými narábame v práci.

Pod biznis procesom (pre účely tejto práce je preň workflow slovo synonymum) rozumieme popis krokov vykonávania úloh, ktoré sú zviazané vzťahmi (napr. podmienkami, cyklami, synchonizáciou a.i.), a ktorý je dostatočne detailný nato, aby mohol byť priamo vykonaný – napr. nejakým podporným softvérom, ktorý voláme **vykonávacie prostredie pre biznis procesy**. Jednotlivé úlohy v biznis procese (vrcholy) a vzťahy medzi nimi (hrany) tvoria orientovaný graf.

Ďalším pojmom, ktorý používame v práci, je **inštancia biznis procesu.** Je ňou ľubovoľné vykonanie konkrétneho biznis procesu. Od biznis procesu sa líši tým, že každá inštancia biznis procesu môže byť iná – ide teda o jedno konkrétne vykonanie biznis procesu. V našom príklade procesu môžu byť napr. nasledovné rôzne inštancie. V jednej inštancii procesu predkladateľ A žiada o nový počítač a jeho nadriadený sa vyjadrí negatívne k správnosti dokumentu, a preto technický správca vráti žiadosť predkladateľovi. V inej inštancii procesu predkladateľ B žiada o novú tlačiareň a jeho nadriadený sa vyjadrí kladne k správnosti dokumentu a táto inštancia procesu pokračuje ďalej od bodu 3.

Súčasťou biznis procesov sú **úlohy**, t. j. typizované jednotky práce a **zdroje** (ľudia, aplikácie, stroje), ktoré úlohy vykonávajú. Podobne ako pri biznis procesoch, aj pri úlohách hovoríme o **inštanciách úloh** – inštanciou rozumieme jedno konkrétne vykonanie úlohy. Navyše úlohy môžeme aj rozdeliť podľa toho, aké zdroje ich vykonávajú na **ľudské úlohy** a **strojové úlohy** (vykonávané strojom, aplikáciou), pretože s každou z týchto skupín úloh sú spojené rozdielne špecifiká. V spomenutom príklade je ľudskou úlohou napr. posúdenie žiadosti nadriadeným, strojovou úlohou v príklade je automatické poslanie informácie predkladateľovi o tom, že sa ňou

1 Pre účely automatizácie tohto biznis procesu sme jeho popis na niektorých miestach v malej miere upravili.

nadriadený v definovanej lehote nezaoberal a zdrojmi sú napr. predkladateľ žiadosti alebo kvestor.

Popíšme teraz motiváciu pre existenciu jazykov na popis procesov.

V organizáciách je bežné, že mnohé z úloh sa vykonávajú opakovane a rutinne – či už strojmi, softvérom alebo ľuďmi. Môže ísť napríklad o automatické zálohovanie dát, vypĺňanie formulárov alebo vymieňanie údajov medzi rôznymi systémami (účtovnícky, personalistický a.i.). Bežnou praxou pri týchto úlohách bolo donedávna ich ručné vykonávanie. V lepšom prípade sú pre konkrétne procesy definované predpísané postupy s podporou zo strany softvéru na ich urýchlenie a zjednodušenie. Problémom však aj za takýchto podmienok ostáva, že nie je možné flexibilne využiť viacero procesov alebo ich častí ako stavebné kamene na to, aby sme z nich mohli vytvoriť iné nové procesy.

Práve pre spomenuté dôvody začalo vznikať viacero jazykov pre popis biznis procesov. Hoci bolo medzi nimi aj niekoľko jazykov (napr. YAWL a BPML), ktoré sa snažili o formálnejšiu sémantiku jazyka a jeho komplexnejší návrh (napr. s podporou rôznych vzorov a aj grafickou notáciou), žiaľ na trhu sa im nepodarilo presadiť. Okrem toho jazyk BPEL – dnes de facto štandard podporuje biznis procesy len z určitej stránky. Procesy v BPEL zapísané možno síce vykonať, avšak pre BPEL nie je definovaná štandardná grafická notácia, čo sťažuje návrh týchto procesov. Preto vznikli aj ďalšie jazyky, ktoré sa úzko zaoberajú práve týmito problémami biznis procesov (BPMN a XPDL).

Pri návrhu spomínaných jazykov (okrem YAWL) sa však väčšinou brali do úvahy ako úlohy, ktoré majú byť súčasťou biznis procesu, iba úlohy vykonávané strojmi alebo programami, resp. úlohy pre ľudí boli súčasťou návrhu jazyk, ale nie v takom ponímaní ako ich používame v našej práci. V našej práci pod úlohami pre ľudí rozumieme úlohy, pri ktorých sa zohľadňujú aj vlastnosti špecifické pre to, že sú vykonávané ľuďmi – napr. ich stavy, možnosť ich delegovania a pod. V reálnej praxi je totižto často potrebné, aby prinajmenšom niektoré z úloh procesu boli vykonávané človekom (rôzne odsúhlasenia, vytvorenie komentárov a pod.). Aj preto vzniklo v júni 2007 rozšírenie BPEL4People pre jazyk BPEL, ktoré adresuje niektoré z problémov týkajúcich sa ľudských úloh v biznis procesoch. V našej práci je ťažiskom práve podpora ľudských úloh v biznis procesoch, najmä v jazyku BPEL.

Motiváciou pre písanie našej práce bolo, že na Univerzite Komenského existuje viacero popisov pre rôzne procesy, ktoré zahŕňajú ako strojové tak aj ľudské úlohy a vznikla potreba vykonávať ich pomocou informačného systému a týmto spôsobom ich urýchliť.

Diplomová práca sa skladá z piatich kapitol a jednej prílohy. **Obsah jednotlivých kapitol a prílohy práce je popísaný v nasledovných odstavcoch.**

V kapitole 1 robíme podrobný prehľad aktuálneho stavu jazykov pre biznis procesy, popisujeme ich možnosti a navzájom ich porovnávame z rôznych hľadísk.

V kapitole 2 predstavujeme iniciatívu *workflow patterns*, ktorá je vedeckou prácou snažiacou sa o vytvorenie vzorov pre rôzne aspekty biznis procesov.

V kapitole 3 popisujeme podrobne podporu ľudských úloh v jazyku BPEL pomocou rozšírenia BPEL4People, analyzujeme podporu pre *workflow resource patterns* v BPEL4People a

robíme prehľad vedeckých prác, ktoré sa doteraz zaoberali podporou ľudských úloh v BPEL.

V kapitole 4 robíme návrh nášho systému, pridávajúceho do BPEL podporu ľudských úloh, na základe vlastností, ktoré vyplývajú z nami vybranej konzistentnej podmnožiny *workflow resource patterns*.

V kapitole 5 zhŕňame výsledky našej práce a zamýšľame sa nad jej ďalším možným rozpracovaním.

V prílohe práce sa nachádza WSDL súbor spolu so súvisiacou XML schémou, ktoré sú použité pri implementácii a popisujú jednotlivé vzory.

V celom texte sú anglické výrazy písané *kurzívou* okrem výrazov, ktoré sa už bežne používajú aj v slovenčine. Anglické výrazy sú ponechané pre lepšie pochopenie textu.

1 Prehľad jazykov pre popis procesov

K jazykom ktoré sú v súčasnosti používané, podporované a vyvíjané v oblasti manažmentu biznis procesov (*Business Process Management*) patria jazyky: Web Services Business Process Execution Language (WS-BPEL), Business Process Modeling Notation (BPMN), XML Process Definition Language (XPDL), Yet Another Workflow Language (YAWL) a iné. Vymenované z nich popíšeme a porovnáme v nasledovných častiach. Na záver posúdime vhodnosť jednotlivých jazykov z hľadiska prípadnej implementácie biznis proces systému na UK.

1.1 Web Services – Business Process Execution Language (WS-BPEL)

Pomocou jazyka WS-BPEL možno definovať rozhrania biznis procesov (teda len popis služby, ktorú ako celok proces ponúka bez konkrétnych detailov, ako sa to v ňom realizuje) ako aj ich vykonateľný popis. Podporuje grafovú štruktúru vzťahov medzi úlohami a je možné ho priamo vykonať. Jeho špecifikácia popisuje voľnou rečou XML syntax a sémantiku jeho prvkov.

1.1.1 História

Tento pomerne nový jazyk (jeho štandardizácia prebehla v roku 2003 neziskovou organizáciou OASIS), je ovplyvnený a historicky vychádza z jazykov XLang a WSFL pre popis procesov, ktorých tvorcami boli firmy Microsoft, resp. IBM. K týmto spoločnostiam sa pridali pri štandardizácii jazyka BPEL aj iní významní hráči ako napr. Oracle, SAP alebo BEA Systems. V súčasnosti sa nachádza vo verzii 2.0 a najmä vďaka širokej podpore významných korporácií a množstva softvérových nástrojov sa považuje za de facto štandard.

1.1.2 Možnosti jazyka

Nasledovné informácie sú čerpané zo špecifikácie jazyka BPEL verzia 2.0 [1]. Jazyk podporuje priamu interakciu iba medzi webovými službami, nie však ľuďmi. Má rozšíriteľnú modulárnu štruktúru založenú na zásuvných moduloch, ktoré sa tvoria ako syntaktické rozšírenia jazyka s popisom sémantiky. Štandardne obsahuje zásuvné moduly pre abstraktné a vykonateľné procesy a pre jazyk XPath.

V jazyku sú použité nasledovné **koncepty a konštrukcie** (v špecifikácii je väčšina z nich nazývaná aktivitami):

- **Pojem procesu** – štandardne sú podporované dva typy procesov:
  - **Abstraktný proces –** Používa sa v obdobnom význame ako rozhranie v klasických programovacích jazykoch v tom zmysle, že sprístupňuje len špecifikáciu činnosti a poskytovaných služieb biznis procesu, avšak bez jeho konkrétnej implementácie. (Takže sa nedá spustiť a vykonať.) Využitie nachádza najmä pri zakrývaní internej programovej logiky a algoritmov (napr. pred konkurenčnými firmami, ktoré využívajú tieto procesy).
  - **Vykonateľný proces –** Je to priama implementácia procesu s detailnými algoritmami, ktorá sa dá vykonať v nejakom vhodnom vykonávacom prostredí.
- **Práca so správami** – Priamo v jazyku sú zabudované možnosti pre prijímanie správ (aktivita `<receive>`) a ich posielanie (aktivita `<reply>`). Tieto možnosti vychádzajú

z jazyka WSDL vytvoreného pre popis webových služieb, ktorý BPEL interne využíva.

- **Volanie webových služieb** sa realizuje aktivitou `<invoke>`. Umožňuje synchrónne ako aj asynchrónne volanie služieb atomickým spôsobom.
- **Aktivity štruktúrovaného programovania** – Patria sem priradenie; podmienky: if-then-elseif-else; cykly: while, repeatUntil, forEach.
- `<sequence>` – Určuje skupinu príkazov, ktoré sa majú vykonať sekvenčne.
- **Paralelizmus** – je podporovaný viacerými spôsobmi:
  - `<forEach>` cyklus – ak je jeho atribút `parallel="yes"`, tak sa paralelne vykonáva n + 1 inštancií[2] popísaných v jeho tele, kde n je rozdiel dolnej a hornej hranice cyklu,
  - `<flow>` – telo tejto aktivity definuje akcie, ktoré majú byť vykonávané súčasne. Pomocou konštrukcie `<links>` je možné definovať závislosť medzi vnorenými akciami,
  - `<pick>` – je aktivita, pri ktorej sa čaká, kým buď nepríde od niektorého z *n* procesov aspoň jedna správa, alebo kým nevyprší vopred stanovená lehota.
- **rozsahy platnosti** (*scope*) – pomocou nich sa dajú definovať lokálne premenné a bloky pre ošetrovanie chýb (aktivitami `<throw>`, `<rethrow>` a `<compensate>`)
- `<validate>` – Je aktivita umožňujúca zistiť, či hodnoty premenných vymenované v jej atribúte `variables` spĺňajú obmedzenia v pridružených XML a WSDL definíciách.

Čo jazyk neumožňuje:

- **Podprocesy** nie sú priamo vo WS-BPEL podporované. Preto ani problémy s nimi spojené (ako napr. automatické ukončovanie dcérskych podprocesov, ak je ukončený rodičovský proces) nie sú v jazyku ošetrené a sú ponechané na programátorov. Existuje však návrh rozšírenia WS-BPEL 2.0 Extensions for Sub-Processes [2] (v spolupráci firiem IBM a SAP), ktoré sa týmto problémom má zaoberať.
- Jazyk nešpecifikuje **syntax pre grafickú vizualizáciu** (ani priamo nepodporuje samotnú vizualizáciu) elementov a činností manažmentu biznis procesov. Touto oblasťou sa zaoberajú iné jazyky ako napr. BPMN (časť 1.2, str. 11) alebo XPDL (časť 1.3, str. 14).

1.1.3 Vykonávacie prostredia pre jazyk WS-BPEL

Ku vykonávacím prostrediam pre jazyk WS-BPEL patria okrem iných aj:

- **S otvoreným kódom:** Apache ODE, ActiveBPEL Engine, Intalio, jBPM (komponent aplikačného servera JBoss)
- **Komerčné:** Oracle BPEL Process Manager, Microsoft Biztalk, Sonic BPEL server, IBM WebSphere Process Server

1.2 Business Process Modeling Notation (BPMN)

Tento jazyk sa používa ako štandardná grafická notácia pre biznis procesy v pracovných tokoch. Hlavným cieľom tvorcov jazyka bolo, aby bol ľahko pochopiteľný pre všetkých, ktorí ho

2 Hoci tu *n* predstavuje hornú hranicu počtu iterácií cyklu resp. počtu inštancií. To, že sa vykonáva n+1 inštancií, sa môže zdať zvláštne, stojí to tak však v špecifikácii BPEL na str. 28.

používajú t. j. od biznis analytikov cez technických vývojárov až po tých, ktorí budú monitorovať a manažovať procesy v ňom graficky zobrazené.

BPMN nie je priamo vykonateľný jazyk, avšak v svojej špecifikácii popisuje podrobne mapovanie na jazyk WS–BPEL. Toto mapovanie je však len čiastočné, keďže BPMN a BPEL nemajú rovnakú výrazovú silu. Samostatne (mimo špecifikácie BPMN) existuje viacero mapovaní z BPMN do vykonateľných jazykov. Zatiaľ najďalej v tejto práci pokročila skupina okolo jazyka XPDL (časť 1.3, str. 14).

1.2.1 História

Jazyk BPMN vznikol v roku 2002 ako vývojová práca firmy IBM. Jeho prvá verzia bola postúpená na štandardizáciu organizácii Business Process Management Initiative (BPMI). Od roku 2005 sa BPMI spojila s organizáciou Object Management Group (OMG), ktorá odvtedy jazyk udržiava a ďalej vyvíja.

1.2.2 Možnosti jazyka

Tieto informácie sú čerpané zo špecifikácie pre jazyk BPMN vo verzii 1.0 [3]. V jazyku sú použité nasledovné **koncepty a konštrukcie**:

- **Pojem procesu** – Štandardne sú podporované tri typy procesov:
  - **Privátny (interný) proces** – Ide o typ procesu, ktorý sa používa v rámci jednej organizácie. Býva nazývaný aj pracovným tokom (*workflow*).
  - **Abstraktný (verejný) proces** – Jeho úlohou je reprezentovať interakcie medzi privátnymi procesmi a iným procesmi alebo účastníkmi. Ostatným účastníkom sprístupňuje len tie správy (prípadne aj s postupnosťami, v akých majú byť vykonané), ktoré sú potrebné k tomu, aby s ním mohli komunikovať.
  - **Kolaboračný (globálny) proces:** – Popisuje interakcie medzi viacerými abstraktnými procesmi.

  Všetky vymenované typy procesov sa v diagramoch môžu spolu vyskytnúť v ľubovoľnej kombinácii.
- **Prvky diagramov** – Tvoria ich nasledovné štyri skupiny: objekty pre popis toku dát a komunikácie, spájajúce objekty, plavecké dráhy a artefakty. Bližšie ich popíšeme v časti 1.2.3, str. 12.
- **Rozšíriteľnosť** – Je možné pridať nové špecializované elementy do jazyka, ktoré však nesmú protirečiť špecifikácii a neprekrývajú sa s grafickou notáciou a sémantikou štandardných prvkov jazyka.

1.2.3 Prvky diagramov

Každý typ prvku má základnú grafickú podobu (napr. kružnica pre udalosť) a podľa svojej špeciálnej úlohy má vnútri priliehavý piktogram (napr. poštovú obálku pre správu alebo hodiny pre časovač). Jednotlivé prvky jazyka sú:

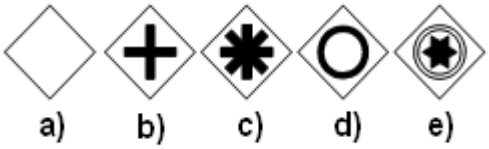
- **objekty pre popis toku dát (*flow objects*):**
  - **udalosť (*event*)** – Popisuje udalosť, ktorá sa stane počas vykonávania biznis procesu. Podľa toho, kedy ovplyvňujú biznis proces, môžu byť udalosti buď štartovné (*start*), vo vykonávaní (*intermediate*) alebo ukončovacie (*end*).

Obrázok 1.1: BPMN - typy udalostí

- **aktivita (*activity*)** – Reprezentuje úkon v biznis procese, ktorý musí byť vykonaný. So znakom + ide o podproces, bez neho o úlohu. (Líšia sa v časovom rozsahu.)

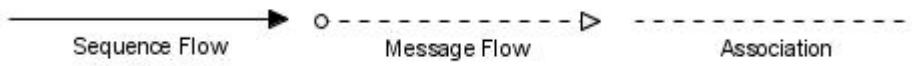
Obrázok 1.2: BPMN - typy aktivít

- **brána (*gateway*)** – Používa sa na rôzne typy rozdelenia a spojenia procesov (najmä synchronizáciu), teda je ňou možné simulovať podmienky. Okrem základných synchronizácii sú podporované aj zložitejšie – napr. xor alebo komplikovanejšie[3].



Obrázok 1.3: BPMN - typy brán:
a) dátovo orientovaná;
b) paralelná (AND); c) komplexná;
d) inkluzívna (OR); e) orientovaná na udalosť

- **spájajúce objekty (*connecting objects*):** v špecifikácii sú zahrnuté ich tri typy a popísané povolené kombinácie:
  - **sekvenčný tok (*sequence flow*)** – Určuje, v akom poradí sa budú vykonávať jednotlivé aktivity.
  - **tok správ** – Definuje, ako sa budú posielať správy medzi procesmi.
  - **asociácia** – Pomocou nej sa k objektu pre popis toku dát môžu pripojiť artefakty, dátové objekty alebo text.



Obrázok 1.4: BPMN - sekvenčný tok, tok správ, asociácia

- **plavecké dráhy** – Používajú sa na logické členenie procesov a služieb a pre určenie viditeľnosti:

3 Príkladom môže byť synchronizácia len prvých troch procesov zo všetkých procesov a podobne.

- **bazén** – Reprezentuje účastníka procesu v pracovnom toku. Kolaboračný proces vypĺňa celý bazén.
- **plavecká dráha** – Používa sa na jemnejšie delenie bazéna. Je vyplnená privátnymi a abstraktnými procesmi.

Obrázok 1.5: BPMN - bazén a plavecké dráhy

- **artefakty** – Sú v jazyku za účelom možnosti pridania doplňujúcich informácií k prvkom diagramu. Je povolené pridávať aj vlastné artefakty (nové, v jazyku ešte nešpecifikované). Nemôžu byť cieľom žiadneho toku riadenia. Patria sem:
  - **dátový objekt** – Môže reprezentovať buď fyzickú entitu (napr. nákladné auto) rovnako ako aj elektronický dokument alebo dátovú štruktúru. Pridáva sa najmä k objektom pre popis toku dát s cieľom podrobnejšie ich popísať. Môže byť vstupom alebo výstupom procesu, ale tiež aj len jeho popisom.
  - **skupina** – Slúži na logické zoskupenie prvkov diagramu bez vplyvu na tok dát a logiku procesov.
  - **anotácia** – Používa sa na bližšie vysvetlenie funkcie a vzťahov medzi elementami diagramu.

Obrázok 1.6: BPMN - dátový objekt, skupina, anotácia

1.2.4 Nedostatky jazyka BPMN

Jednotlivé pojmy v sémantike jazyka sú definované neformálne, len v prirodzenom jazyku, čo môže viesť k nekompatibilite a rôznej interpretácii v jednotlivých nástrojoch. K formátu ukladania prvkov jazyka BPMN sa špecifikácia nevyjadruje vôbec. Túto úlohu medzičasom na seba prevzal jazyk XPDL (pozri časť 1.3, str. 14) a popísané nedostatky jazyka by mali byť odstránené v pripravovanej verzii 2.0 [4]. Jazyk sa nedá priamo vykonať.

1.3 XML Process Definition Language (XPDL)

Jazyk XPDL si dáva v aktuálnej verzii 2.0 za cieľ byť najmä prenositeľným formátom, ktorý uchováva informácie o biznis procesoch. Tvorcovia jazyka tvrdia, že prenositeľnosť informácií, ktoré sú štandardnou súčasťou špecifikácie jazyka [5] (teda nejde o rozšírenia) medzi jednotlivými nástrojmi, ktoré spĺňajú špecifikáciu XPDL, by mala byť takmer stopercentná.

Podľa tvorcov jazyka a tiež článku [6] existuje bijekcia medzi jazykmi XPDL a BPMN.

Vďaka tomu je jazyk XPDL vhodný na prenos diagramov medzi jednotlivými nástrojmi[4]. Aj z tohoto dôvodu jeho nová verzia 2.0 doplnila v špecifikácii najmä prvky, ktoré boli súčasťou jazyka BPMN, ale chýbali v jazyku XPDL.

Jazyk je spätne kompatibilný so svojou predchádzajúcou verziou 1.0 s výnimkou zápisu rozsahov platnosti identifikátorov (*namespaces*).

1.3.1 Možnosti jazyka

Keďže jedným z cieľov jazyka XPDL je, aby bol formátom na ukladanie informácií o BPMN diagramoch, tak podporuje všetky konštrukcie, ktoré sa vyskytujú aj v jazyku BPMN, a preto ho tu nebudeme podrobnejšie popisovať. (Pre jednotlivé konštrukcie pozri časť 1.2.2, str. 12).

1.4 Yet Another Workflow Language (YAWL)

1.4.1 História

Vývoj jazyka YAWL začal v roku 1999. Najdôležitejším dôvodom pre jeho vznik bola neexistencia jazyka – štandardu pre popis biznis procesov. Žiaden z vtedajších jazykov nepodporoval všetky vzory pre biznis procesy (pozri časť 1.5, str. 18) a len máloktoré mali definovanú sémantiku (a ak aj mali, tak zväčša neformálne). Tvorcovia tohto jazyka si dali za cieľ odstrániť všetky tieto nedostatky [7].

Vývoj jazyka bol zo začiatku čisto akademickou prácou dvoch univerzít – výskumnej skupiny zaoberajúcej sa BPM na austrálskej Technickej univerzite v štáte Queensland (http://www.bpm.fit.qut.edu.au/) a holandskej Technickej univerzity v Eindhovene (http://w3.tue.nl/en/). Medzičasom začalo jazyk používať a prispievať k jeho vývoju aj niekoľko stredne veľkých firiem – napr. medzinárodná skupina hotelov InterContinental Hotels Group.

1.4.2 Možnosti jazyka

Pre tvorcov bol jeden z najdôležitejších cieľov, aby bol jazyk YAWL schopný podporovať čo najviac zo vzorov pre pracovné toky (*workflow patterns*), ktoré vypracovala skupina workflow patterns initiative [8]. Tieto vzory vznikli po dôkladnej analýze množstva rôznych diagramov pre biznis procesy. Sú vhodné pre porovnávanie jazykov z hľadiska možností, ktoré poskytujú v oblasti pracovných tokov [9].

Aby sa v sémantike jazyka dosiahli uvedené ciele, bol jeho formálny model inšpirovaný modelom Petriho sietí. **Petriho siete** sú matematickým modelom popisujúcim paralelné procesy. Tento model však niektoré zo vzorov pre pracovné toky nepodporuje (ide o skupinu vzorov pre ukončovanie a rušenie, vzor pre synchronizované spojenie a skupinu vzorov týkajúcich sa viacerých inštancií). Jazyk YAWL však podporuje takmer všetky vzory pre pracovné toky, a teda aj jeho model je širší ako pri Petriho sieťach. YAWL ale nie je rozšírením Petriho sietí. Je úplne novým jazykom s formálnym matematickým modelom, ktorý bol iba inšpirovaný Petriho sieťami ako aj konceptami zo vzorov pre pracovné toky.

Pre mnohé svoje koncepty má jazyk aj grafickú reprezentáciu, ktorú v ďalšom podrobnejšie popíšeme. Jazyk je možné aj priamo vykonať. Podobne ako ostatné jazyky z oblasti BPM aj YAWL ukladá údaje o diagramoch v XML súbore, popísanom pomocou XML schémy.

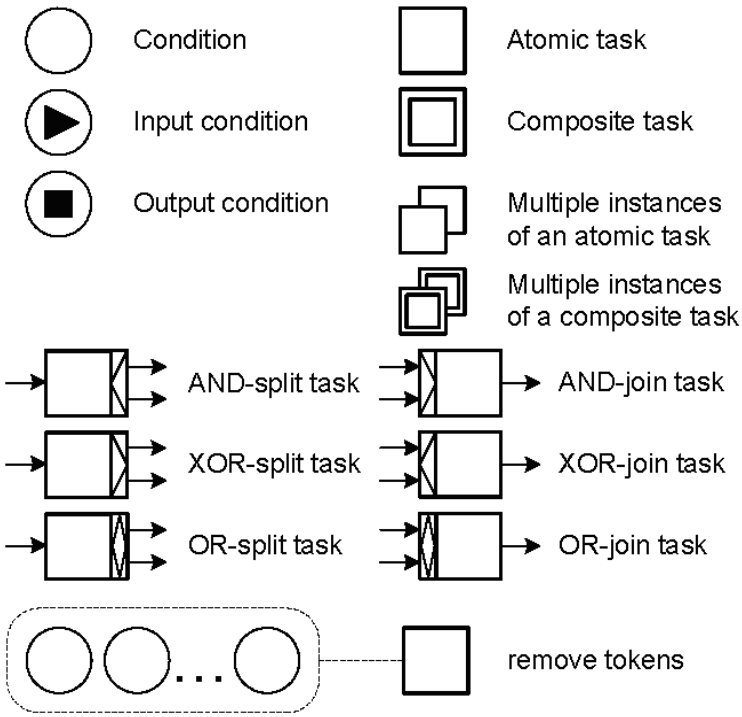
4 Keďže zo sémantického hľadiska nie sú jazyky WS-BPEL a XPDL (a teda aj BPMN, pretože BPMN a XPDL majú rovnakú výrazovú silu) navzájom podmnožinami, neexistuje medzi nimi bijekcia. Možné sú len injekcie z XPDL (BPMN) do WS-BPEL.Teda nie všetky konštrukcie zapísané v jazyku XPDL resp. BPMN možno preložiť do BPEL.

1.4.3 Prvky diagramov

S jednotlivými prvkami diagramov sú spojené nasledovné pojmy:

- **pracovný tok** – Je v YAWL definovaný ako množina definícií procesov tvoriacich stromovú hierarchiu. V hierarchii existuje jeden koreňový pracovný tok.
- **úloha** (*task*) – môže byť buď *atomická,* alebo *zložená*. Atomické úlohy tvoria v stromovej štruktúre listy, zložené úlohy sú tvorené buď opäť zloženými úlohami, alebo atomickými úlohami.

Na nasledovnom obrázku sú tie prvky jazyka, ktoré majú aj grafickú reprezentáciu.



Obrázok 1.7: YAWL - grafické symboly jazyka.
Obrázok je prevzatý z [7], str. 4.

Na obrázku 1.7 je možné vidieť, že jazyk YAWL podporuje existenciu viacerých inštancií rovnakej úlohy (pomocou konštrukcií *Multiple instances of an atomic task* a *Multiple instances of a composite task*). V najspodnejšej časti obrázku sa nachádza možnosť odstrániť prvky diagramu (*remove tokens*). Vďaka tejto konštrukcii YAWL podporuje vzory pre pracovné toky týkajúce sa ukončovania a rušenia (*cancellation and force completion patterns*).

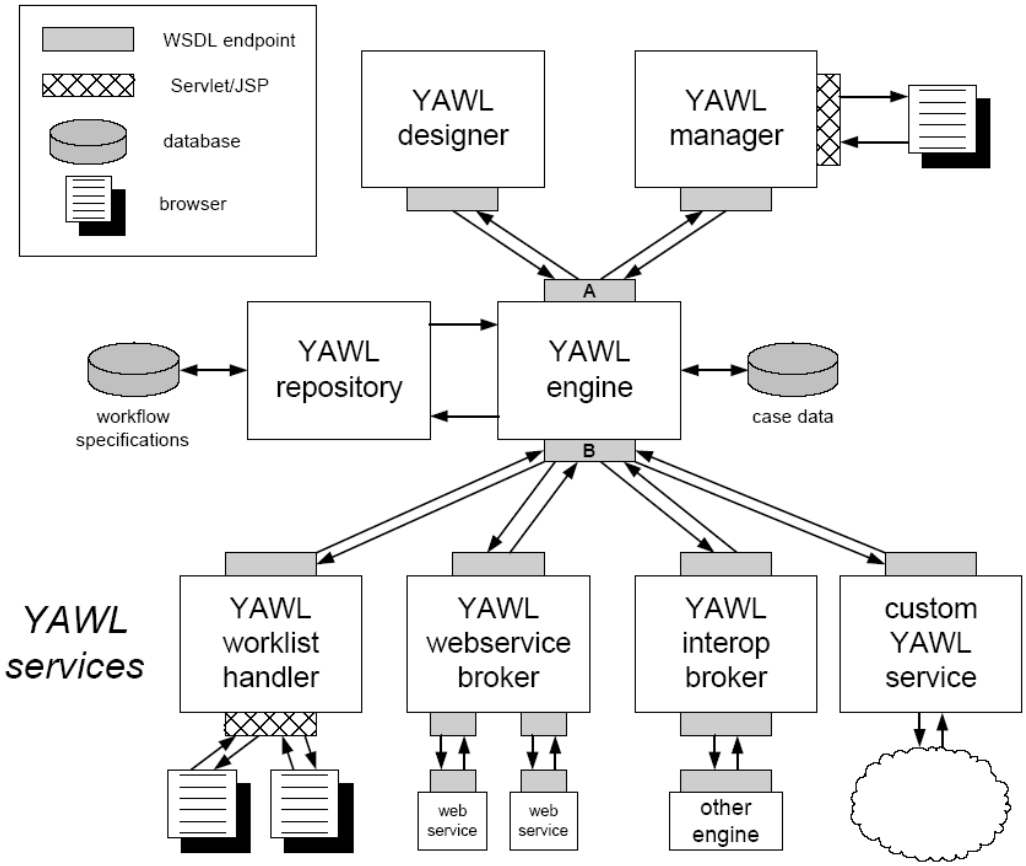
1.4.4 Architektúra systému YAWL

Krátko ešte popíšme niektoré zaujímavé veci z architektúry YAWL systému. Architektúra systému je modulárna a skladá sa z nasledovných prvkov. (Schematicky sú znázornené aj na obr. 1.8)

- **jadro** (*engine*) pre vykonávanie kódu zapísaného v jazyku YAWL
- **editor YAWL** pre grafický návrh diagramov
- skupina tzv. **služieb YAWL**, kam patria:

- **YAWL worklist handler** – priraďuje prácu jednotlivým používateľom systému. V YAWL táto služba nebola zakomponovaná do jadra, vďaka čomu môže byť jadro navrhnuté výpočtovo čo najefektívnejšie,
- **YAWL webservice broker** – má na starosti obojsmernú komunikáciu medzi webovými službami a jadrom YAWL systému,
- **ostatné služby**: – YAWL interop broker, custom YAWL service – možnosť pridania vlastného modulu do YAWL systému spĺňajúceho špecifické požiadavky jeho tvorcov.

- **manažér YAWL** – Webové rozhranie pre konfiguráciu systému YAWL
- **YAWL repository** – Komponent pre správu inštancií pracovných tokov



Obrázok 1.8: YAWL - architektúra systému prevzatý z [7], str. 8

1.4.5 Vývojové prostredia pre YAWL

V čase písania práce existuje pre YAWL iba prostredie, ktoré vytvorili samotní dizajnéri jazyka nachádzajúce sa vo verzii beta 8 [10] .

1.5 Porovnanie jazykov pre popis procesov

1.5.1 Porovnanie z hľadiska problémových oblastí, ktoré pokrývajú

V tejto časti zhrnieme už spomenuté skutočnosti v predchádzajúcich častiach kapitoly 1 a doplníme ich ďalšími informáciami.

WS-BPEL je jazyk používajúci štruktúru orientovaného grafu (čo sa vzťahov medzi procesmi týka – napr. pri posielaní správ alebo toku dát) ako aj štrukturálne konštrukcie (podmienky, cykly, a.i). Je možné ho priamo vykonať vo vhodnom vykonávacom prostredí. Je široko podporovaný, vďaka čomu ho možno považovať za de facto štandard.

BPMN je jazyk na popis grafickej notácie biznis procesov, tvorených v grafickom režime. Nie je priamo vykonateľný. Je síce možná čiastočná transformácia z BPMN do WS-BPEL kódu, avšak niektoré konštrukcie sa z BPMN do BPEL nedajú preložiť.

XPDL je určený na popis BPMN diagramov. V jazyku však nie je špecifikovaný grafický vzhľad jeho prvkov, ukladajú sa len informácie o nich. Preto je jazyk vhodný ako medziformát pre bezstratový prenos informácií o BPMN diagramoch medzi jednotlivými modelovacími nástrojmi.

YAWL je jazyk, ktorý má výrazovú silu porovnateľnú s WS-BPEL. Je priamo vykonateľný, definuje grafickú podobu svojich prvkov a má aj vlastný XML formát, do ktorého ukladá informácie o diagramoch. Okrem toho má formálne – matematicky definovanú sémantiku. Tento jazyk preto z teoretického hľadiska možno považovať za priameho konkurenta predchádzajúcej trojice jazykov. V praxi však tomu tak nie je, najmä pre malú podporu na trhu.

1.5.2 Porovnanie z hľadiska funkčných aspektov, aspektov správania sa a informačných aspektov

Pretože z hľadiska popisnej sily existuje medzi jazykmi XPDL a BPMN bijekcia (pozri [6]), stačí nám ďalej porovnávať len jazyky WS-BPEL, BPMN a YAWL. Porovnanie urobíme na základe informácií z článku [11].

Funkčné aspekty sa sústreďujú v biznis procese na to, čo sa má vykonať a zároveň určujú hierarchické vzťahy medzi úlohami a podúlohami. K funkčným aspektom patria: aspekt kompozície pracovných tokov (*nesting*) a *aspekt obmedzení* kladených na pracovné toky.

Aspekty týkajúce sa správania popisujú riadenie postupnosti vykonávania úloh v procese.

Informačné aspekty sa sústreďujú na poskytnutie relevantných údajov systémom vo vhodnom čase.

Skupiny aspektov a ich podporu jednotlivými jazykmi vysvetlíme pomocou tabuľky 1.1. Zameriame sa najmä na tie aspekty, s ktorými majú jednotlivé jazyky problém a vysvetlíme v čom.

Ako je z tabuľky vidieť, žiaden z jazykov nemá problémy so skupinou **funkčných aspektov** a ani s podskupinou **základné typy vetvenia a synchronizácie**.

	BPEL	BPMN	YAWL
funkčné aspekty			
hierarchie	X	X	X
obmedzenia	X	X	X
aspekty týkajúce sa správania			
základné typy vetvenia a synchronizácie	X	X	X
pokročilé vetvenie a synchronizácia	–	X	X
štrukturálne aspekty	O	X	X*
viacnásobné inštancie	X	O	X
úlohy ovplyvnené inými úlohami	X*	X*	X
informačné aspekty			
viditeľnosť, interakcia, smerovanie	O	O	–*

Tabuľka 1.1: Porovnanie jazykov BPEL, BPMN a YAWL prevzatá z [11]

Vysvetlivky: **X** – plne podporované aspekty, **O –** je možná okľuka, X* – aspekty podporované z väčšej časti, –* aspekty podporované z menšej časti, – nepodporované aspekty

S podskupinou **pokročilé vetvenie a synchronizácia** má problémy iba jazyk BPEL – podporuje z nich iba vzor pre výber z viacerých možností. Ďalšou skupinou sú **štrukturálne aspekty** týkajúce sa najmä cyklov a ukončovania procesov. BPEL sa neumožňuje do cyklov dostať ľubovoľným spôsobom. Preto nepodporuje vzor pre ľubovoľný cyklus[5]. Explicitné ukončovanie funguje tak, že príkaz `flow` skončí, keď skončia všetky aktivity definované v jeho tele. YAWL nemá pre vzor implicitného ukončovania vhodný prostriedok. Je ale možné docieliť podobný výsledok spojením všetkých koncových úloh pomocou OR spojenia. S podskupinou **viacnásobné inštancie** má menší problém len jazyk BPMN. Nedá sa v ňom vytvoriť viacero nesynchronizovaných inštancií úloh. S viacerými inštanciami bez vedomosti o ich počte pred vykonávaním v BPMN možno pracovať len okľukou (*workaround*). V skupine aspektov, kde **sú úlohy ovplyvnené inými úlohami**, je problém v jazykoch BPEL a BPMN iba s tým, že nepodporujú vzor míľnik[6], no v oboch jazykoch sa dá realizovať okľukou.

Informačné aspekty sa zaoberajú údajmi, ktoré sú spracovávané biznis proces systémom. My sa budeme zaujímať o: viditeľnosť dát, interakciu medzi dátami, prenos a smerovanie dát. S **viditeľnosťou dát** nie je v žiadnom z jazykov problém. Čo sa **interakcie medzi dátami** týka, tak BPEL ani YAWL s touto skupinou nemajú väčšie problémy, avšak v **BPMN** je viditeľnosť podporovaná len na lokálnej úrovni t. j. problémová je interakcia s údajmi, ktoré sú externé vzhľadom na biznis proces (napr. údaje v biznis procese z externej inštitúcie). S poslednou podskupinou – **smerovanie dát** má problém len jazyk BPEL. Konkrétne má problém s úlohami, ktorých spúšťač je založený na dátach. Tie nie sú podporované priamo, no existuje pre ne okľuka.

5 Ľubovoľným cyklom sa tu myslí cyklus, ktorý by v procedurálnych programovacích jazykoch zodpovedal cyklu vytvorenému pomocou goto príkazov.

6 Aktivita je umožnená, iba ak je inštancia procesu v určitom stave vykonávania – míľniku. Ak sa proces dostane za míľnik, aktivitu s ním spojenú už nemožno nikdy vykonať.

1.5.3 Výber jazyka pre implementačné účely na UK

Na základe prehľadu a porovnania jazykov v celej kapitole 1 sme sa rozhodli pre popis biznis procesov v našej implementácii vybrať jazyk BPEL z nasledovných dôvodov:

1. **YAWL** by bol ideálny jazyk z teoretickej stránky, pretože podporuje takmer všetky *workflow patterns*, má formálnu matematickú sémantiku, je vykonateľný, má aj grafickú reprezentáciu a nástroj pre prácu s ním má otvorený zdrojový kód. K jeho nedostatkom patrí predovšetkým malá rozšírenosť ale aj to, že jediné softvérové prostredie pre prácu s ním bolo v čase písania tejto práce len vo verzii beta. Z týchto dôvodov YAWL z ďalšieho výberu vylúčime.

2. **BPMN** je jazyk určený najmä na grafický popis biznis procesov. Nie je priamo vykonateľný, no popisuje čiastočne preklad z BPMN do BPEL. Navyše nepodporuje prácu s ľudskými úlohami, resp. nie v takom ponímaní, ako ich chápeme v našej práci (teda ako úlohy, pri ktorých sa zohľadňujú aj vlastnosti špecifické preto, že sú vykonávané ľuďmi – napr. ich stavy, možnosť ich delegovania a pod.). Pre tieto dôvody BPMN z výberu vylúčime.

3. Keďže jazyk **XPDL** je len ukladacím formátom pre BPMN a má identickú výrazovú silu ako on, tak tento jazyk z výberu rovnako môžeme vylúčiť.

4. Najhlavnejší dôvod hovoriaci pre jazyk **BPEL** bol, že UK zakúpila počas písania práce softvér na vytváranie a manažment biznis procesov vytváraných práve v BPEL. Okrem toho má BPEL za sebou aj podporu veľkých firiem a je dostatočne používaný, vďaka čomu ho je možno považovať za de facto štandard. Je priamo vykonateľný a jeho sémantika je popísaná formálnejšie ako pri BPMN, čo zvyšuje kompatibilitu medzi nástrojmi. Tiež podporuje prácu s ľudskými úlohami, hoci iba pomocou rozšírenia BPEL4People (pozri časť 3.1, str. 24).

2 Workflow patterns

Workflow patterns initiative [8] je akademická iniciatíva snažiaca sa o identifikovanie najčastejších opakujúcich sa postupov pri návrhu biznis procesov a o ich spracovanie vo forme vzorov (*pattern*). Zaoberá sa formálnymi ako aj implementačnými problémami súvisiacimi so vzormi. Táto práca bola započatá v roku 1999 a je vedená najmä dvoma vedeckými tímami: tímom na holandskej Eindhoven University of Technology (okolo prof. Wil van der Aalst-a) v spolupráci s tímom na austrálskej Queensland University of Technology (okolo prof. Arthur ter Hofstede-ho). Jednotlivé vzory sú rozdelené do viacerých logických skupín, ktoré sú stručne popísané v nasledovnej časti.

2.1 Rozdelenie vzorov

Pri skupinách uvádzame pôvodné anglické termíny, aby ich bolo možné v prípade čitateľovho záujmu ľahšie nájsť. Ak by sme spravili ich preklad do slovenčiny (ktorý je niekedy v tejto oblasti trochu ťarbavý) bolo by to sťažené. Pri každej skupine buď popíšeme, čo vo všeobecnosti zahŕňa, alebo pre urobenie predstavy vymenujeme niektoré konkrétne vzory. Keďže všetkých vzorov je vyše 100 a často sa líšia v maličkostiach, bolo by príliš rozsiahle ich v našej práci pre úplné pochopenie dopodrobna opisovať. Čitateľovi, ktorý má o ne hlbší záujem, odporúčame ich domovskú webovú stránku [8].

Vzory pre riadenie (*control-flow patterns*)

Tieto vzory boli prvým vedeckým výsledkom *workflow patterns initiative*. Popisujú rôzne aspekty riadenia vykonávaného v biznis procesoch.

Táto skupina má nasledovné podskupiny:

- basic control flow patterns – Popisujú sekvenčné vykonávanie úloh, paralelné rozdelenie úlohy a.i.
- advanced branching and synchronization patterns – Popisujú synchronizáciu a delenie úloh s ohľadom na rôzne podmienky.

  niektoré vzory: *1 out of m join* – z *m* vstupov bude aktívny iba *1* výstup – ten zo vstupov, ktorý bol najskôr aktivovaný; *multi-choice* – z *n* vstupov a *k* výstupov sa vyberie *r* výstupov, ktoré budú aktívne.
- multiple instance patterns – Popisujú vytváranie a synchronizáciu viacerých úloh v rôznych variantoch.
- state-based patterns – Popisujú situácie, ktoré závisia od stavu celého biznis procesu alebo od časového momentu, v ktorom sa aktuálna úloha vykonáva.

  niektoré vzory: deferred choice, interleaved parallel routing
- cancellation and force completion patterns – Popisujú odstránenie úlohy, skupiny úloh, celej inštancie procesu a pod.
- iteration patterns – Popisujú cyklické a rekurzívne vykonávanie úloh.
- termination patterns – Popisujú okolnosti, kedy biznis proces môžeme považovať za ukončený.
- trigger patterns – Popisujú externé udalosti, ktoré môžu byť potrebné, aby sa úloha mohla iniciovať.

Vzory týkajúce sa dát

Dátami sa tu myslia ľubovoľné dáta – či už primitívne (napr. boolean), alebo štruktúrované (napr. pole, dokument).

Táto skupina má nasledovné podskupiny:

- data visibility – Popisujú rôzne rozsahy viditeľnosti dát – dáta viditeľné pre: jednu úlohu, pre jednu inštanciu biznis procesu, pre všetky úlohy v podprocese a.i.
- data interaction – Popisujú, ako sa presúvajú dáta v rámci rôznych blokov jednej inštancie biznis procesu (inej úlohe, podprocesu atď.), alebo v rámci rôznych blokov viacerých biznis procesov.
- data transfer patterns – Popisujú rôzne mechanizmy, ako sa fyzicky presúvajú dáta medzi komponentami v rámci jedného procesu. Patria sem mechanizmy presúvania premenných hodnotou, kópiou, referenciou, alebo ich transformáciou tesne pred presunutím.
- data-based routing patterns – popisujú, ako sa mení riadenie v biznis procese závisiace buď od existencie určitých dát alebo ich hodnotách vyprodukovaných inými úlohami. Patria sem aj vzory popisujúce vonkajšie udalosti, ktoré posielajú aktuálnej úlohe dáta.

Vzory pre ošetrovanie výnimiek

Táto skupina sa zaoberá všeobecnými charakteristikami chybových stavov na vyššej úrovni abstrakcie – na úrovni biznis procesov a ich logiky, nie na nižšej úrovni (napr. delenie nulou).

Na typ odpovede pri zotavení sa z výnimky sa dá nazerať ako na **vzor** stručne popisujúci spôsob zotavenia sa z výnimky, ktorý bude vykonaný. **Vzor pre ošetrovanie výnimiek** má formu n-tice pozostávajúcu z nasledovných častí:

- Ako sa má vykonávacie prostredie pre biznis proces správať k úlohe, ktorej sa výnimka týka.
- Ako sa má vykonávacie prostredie pre biznis proces správať k inštancii biznis procesu, ktorej sa výnimka týka.
- Akú akciu pre zotavenie sa z výnimky je potrebné vykonať.

Vzory týkajúce sa zdrojov

Táto skupina má nasledovné podskupiny: creation patterns, push patterns, pull patterns, detour patterns, auto-start patterns, visibility patterns, multiple resource patterns. Podrobne je popísaná v nasledovnej časti 2.2, keďže jedným z cieľov našej implementácie bolo, aby podporovala konzistentnú z nich vybranú podmnožinu.

2.2 Workflow Resource Patterns

Zdrojom sa pri workflow resource patterns myslí ľubovoľná entita schopná vykonať prácu. Najčastejšie ide o človeka, ale môže ňou byť napr aj aplikácia, webová služba, stroj, procesor a.i. V nasledovných častiach popíšeme podrobne jednotlivé skupiny workflow resource patterns. Čísla pred jednotlivými vzormi uvádzame preto, aby ich čitateľ v prípade záujmu vedel ľahšie nájsť na stránke [8], kde sú tiež číslované.

2.2.1 Creation Patterns

Popis: Tieto vzory popisujú rôzne obmedzenia, za ktorých môžu byť úlohy vykonané. Obmedzenia sú definované v čase návrhu biznis procesu a ovplyvňujú spôsob, akým sa úlohy prideľujú zdrojom.

Vzory v tejto skupine: 1) Direct Allocation, 2) Role-Based Allocation, 3) Deferred Allocation, 4) Authorisation, 5) Separation of Duties, 6) Case Handling, 7) Retain Familiar, 8) Capability-Based Distribution, 9) History-Based Distribution, 10) Organisational Distribution, 11) Automatic Execution

2.2.2 Push Patterns

Popis: Táto skupina sa zameriava na situácie, keď *vykonávacie prostredie pre biznis proces* proaktívne ponúka, alebo prideľuje novovytvorené úlohy zdrojom. Toto môže byť robené priamo i nepriamo. Prvé tri vzory definujú, či sa úlohy ponúknu jednému alebo viacerým zdrojom, resp. či sa zdroju priamo pridelia. Vzory 15 – 17 popisujú algoritmus, na základe ktorého je vybraný zdroj na spracovanie nasledovnej úlohy spomedzi viacerých potenciálnych kandidátov. Posledná skupina vzorov určuje načasovanie distribúcie úloh zdrojom – obzvlášť vzťah dostupnosti úlohy na ponúknutie resp. pridelenie a času, kedy na nich zdroje začnú pracovať.

Vzory v tejto skupine: 12) Distribution by Offer – Single Resource, 13) Distribution by Offer – Multiple Resources, 14) Distribution by Allocation – Single Resource, 15) Random Allocation, 16) Round Robin Allocation, 17) Shortest Queue, 18) Early Distribution, 19) Distribution on Enablement, 20) Late Distribution

2.2.3 Pull Patterns

Popis: Popisujú situácie, keď sú zdroje nejakým spôsobom upovedomené o úlohách, ktoré možno vykonať. Záväzok vykonať konkrétnu úlohu robí zdroj a nie vykonávacie prostredie pre biznis proces. Prvé tri vzory definujú špecifiká konkrétnej akcie z tejto skupiny sústreďujúc sa najmä na stavy pred a po vykonaní akcie. Ďalšie vzory tejto skupiny sa zameriavajú na postupnosť, v akej sú úlohy zdroju prezentované a ako túto postupnosť môže ovplyvniť vykonávacie prostredie pre biznis proces a ako človek. Posledný vzor tejto skupiny opisuje stupeň slobody, ktorý má zdroj pri výbere ďalšej úlohy.

Vzory v tejto skupine: 21) Resource-Initiated Allocation, 22) Resource-Initiated Execution – Allocated Work Item, 23) Resource-Initiated Execution – Offered Work Item, 24) System-Determined Work Queue Content, 25) Resource-Determined Work Queue Content, 26) Selection Autonomy

2.2.4 Detour Patterns

Popis: Tieto vzory sa vzťahujú na prípady, kedy vykonávanie už pridelených úloh zdrojom je prerušené buď zo strany vykonávacieho prostredia pre biznis proces alebo na podnet zdroja.

Vzory v tejto skupine: 27) Delegation, 28) Escalation, 29) Deallocation, 30) Stateful Reallocation, 31) Stateless Reallocation, 32) Suspension/Resumption, 33) Skip, 34) Redo, 35) Pre-Do

2.2.5 Auto-Start Patterns

Popis: Vzťahujú sa na situácie, keď vykonávanie úloh je iniciované špecifickými udalosťami v životnom cykle úlohy, alebo vyplýva z definície biznis procesu. K týmto udalostiam patria napr. vytvorenie a pridelenie úlohy, ukončenie predošlej alebo inej úlohy.

Vzory v tejto skupine: 36) Commencement on Creation, 37) Commencement on Allocation, 38) Piled Execution, 39) Chained Execution

2.2.6 Visibility Patterns

Popis: Táto skupina vzorov zatrieďuje rozsahy viditeľnosti pre dostupnosť a stav úloh v celom systéme z pohľadu zdrojov.

Vzory v tejto skupine: 40) Configurable Unallocated Work Item Visibility, 41) Configurable Allocated Work Item Visibility

2.2.7 Multiple Resource Patterns

Popis: Tieto dva vzory popisujú paralelné vykonávanie viacerých úloh jedným zdrojom, resp. prácu viacerých zdrojov na jednej úlohe.

Vzory v tejto skupine: 42) Simultaneous Execution, 43) Additional Resources

3 Podpora úloh pre ľudí v BPEL

Poznámka: V tejto kapitole sú slová zdroj a človek chápané ako ekvivalent.

V aktuálnej verzii špecifikácie jazyka WS-BPEL 2.0 neexistuje priama podpora práce pre ľudí (skrátene ich budeme nazývať ľudské úlohy). Keďže v takmer každom biznis procese do nejakej miery participujú aj ľudia (napr. potvrdenie pracovnej cesty) s rôznymi úlohami, cieľmi a právami, je absencia priamej podpory ľudských úloh závažným nedostatkom jazyka BPEL.

V štandardnom BPEL sa dá použiť iba zavolanie webovej služby, za ktorou môže byť skrytá vhodná aplikácia umožňujúca zapojenie človeka do biznis procesu. **Tento prístup má viacero nedostatkov**:

1. **Nie je štandardizovaný**. Preto viacero vytvorených procesov (najmä v medzifiremnom prostredí) bude navzájom nekompatibilných, čo sa ľudských úloh týka. Toto je v dobe častej participácie na projektoch na medziorganizačnej alebo medzifiremnej úrovni prinajmenšom veľmi časovo a finančne náročné , resp. neakceptovateľné.

2. **Je ťažké, ak je to vôbec možné, udržiavať konzistentne informácie o stave úlohy a jej ostatných atribútoch** (napr. lehoty pri vzore 28) Escalation). Z toho vyplývajú aj komplikované riešenia prispôsobené len na konkrétny problém. Tiež je náročné zaručiť dodržiavanie pokročilejších vzorov z workflow resource patterns.

Pre tieto dôvody popíšeme v ďalšej časti, aké práce boli doteraz vykonané v ohľade zakomponovania ľudských úloh do BPEL a následne navrhneme aj vlastné riešenie v kapitole 4.

3.1 BPEL4People

BPEL4People je špecifikácia vydaná v júni 2007 vytvorená v spolupráci viacerých veľkých softvérových spoločností (IBM, Oracle, SAP, Adobe, BEA a Active Endpoints). Jej vydaniu predchádzal v roku 2005 koncept špecifikácie. Špecifikácia sa skladá z dvoch dokumentov: *Web Services Human Task (WS-HumanTask)* [12] a *WS-BPEL Extension for People (BPEL4People)* [13]. Obe špecifikácie popisujú XML schému pridaných syntaktických prvkov do jazyka BPEL. Okrem toho slovne definujú ďalšie podmienky ako aj sémantiku pridaných prvkov.

3.1.1 Špecifikácia WS-HumanTask

Táto špecifikácia rozširuje jazyk BPEL najmä o nasledovné vlastnosti a možnosti:

- Pojem úlohy určenej pre ľudí (*human task*) s popismi jej možných vlastností a operácií, ktoré možno s nimi vykonávať.
- Pojem notifikácie. Notifikácia je úloha určená na odovzdanie nejakej informácie zdroju, ktorá nikdy nevyžaduje odpoveď. Teda má pre zdroje len informatívny charakter.
- Niektoré z rozširujúcich operácií: eskalovanie, nominácia, reťazové vykonávanie viacerých úloh a.i.
- Prácu s úlohami na základe rolí. (Viacero základných rolí je preddefinovaných.)
- Spôsob, ako priradiť ku generickej úlohe konkrétneho človeka.
- Stavový stroj (*state machine*) s popismi stavov a podmienkami, na základe ktorých môže úloha prechádzať medzi jednotlivými stavmi.
- Definuje štandardné operácie pre zmeny stavov, rozširujúce operácie a s nimi súvisiace výnimky.

- Prílohy, ktoré môžu byť ľubovoľného typu, ktorý je podporovaný XML schémou použitou v zápise BPEL procesu, resp. ľubovoľného MIME typu dokumentu. Takto možno k úlohám pripojiť napríklad formuláre na vyplnenie, vysvetľujúce dokumenty a pod.
- Atribúty (napr. stav úlohy, čas vytvorenia, priorita), ktoré musí každá úloha pre ľudí podporovať a možnosť vyhľadať úlohy pomocou dotazov nad atribútmi.

3.1.2 Špecifikácia BPEL4People

Špecifikácia BPEL4People je realizovaná ako rozšírenie (*extension*) jazyka BPEL. Do BPEL pridáva novú aktivitu `<peopleActivity>`, ktorá je implementovaná pomocou *human task* definovanej v predošlej časti 3.1.1. Do jazyka BPEL a špecifikácie WS-HumanTask pridáva nasledovné vlastnosti a možnosti:

- priradenie konkrétnych ľudí alebo rolí k úlohám buď pomocou rozšírenia štandardnej BPEL konštrukcie `<assign>` alebo pomocou dotazov,
- úlohy s rôznou viditeľnosťou – lokálne sú viditeľné vrámci jedného BPEL procesu; verejné pre všetky BPEL procesy pristupujúce k nim cez rozhranie webovej služby,
- možnosti pridať k úlohám atribúty oneskorenia (*defer;* napr. časový interval, dokedy možno posunúť vykonávanie úlohy) a času platnosti úlohy (*expiration*),
- viacero ďalších menších rozširujúcich atribútov oproti špecifikácii WS-HumanTask.

3.1.3 BPEL4People v porovnaní s workflow resource patterns

V tejto časti sme na základe porovnania a analýzy informácií zo zdrojov stránky *workflow patterns initiative* [8], špecifikácie Web Services Human Task (WS-HT) [12] a špecifikácie WS-BPEL Extension for People (BPEL4people) [13] vytvorili **vlastné porovnanie podpory workflow resource vzorov dvojicou špecifikácií** [12] **a** [13] **uvedené v nasledovnej tabuľke**. V stĺpci poznámky sú buď odkazy na časti dokumentov, kde sa o funkčnosti vyžadovanej daným vzorom píše, alebo krátke odôvodnenie, prečo konkrétny vzor považujeme za podporovaný, resp. nepodporovaný.

skupina vzorov	vzor	podpora	poznámky
creation patterns	1) Direct Allocation	X	Pozri časť 3.2.2 vo WS-HT – Using Literals.
	2) Role-Based Allocation	X	Pozri časť 3.2.1 vo WS-HT – Using Logical People Groups.
	3) Deferred Allocation	X	Pozri časť 3.2.3 vo WS-HT – Using Expressions (for assigning people).
	4) Authorisation	-	Autorizácia v týchto vzoroch znamená, že daný zdroj musí spĺňať nejakú podmienku. Toto nie je pre všeobecnú podmienku možné dosiahnuť štandardnými prostriedkami popísanými v špecifikáciách BPEL 2.0, BPEL4people a WS-HT.
	5) Separation of Duties	X	Pozri časť 6.2 XPath Extension Functions vo WS-HT – funkcia except. V náčrte špecifikácie BPEL4people je pojem obsiahnutý v tomto vzore nazvaný aj

skupina vzorov	vzor	podpora	poznámky
			„princíp štyroch očí“.
	6) Case Handling	-	Tento vzor používa len zoslabené podmienky vzoru 7) a teda je len jeho špeciálnym prípadom. Preto sa naň dá aplikovať rovnaký dôvod, prečo nie je podporovaný.
	7) Retain Familiar	-	V BPEL4people neexistuje všeobecný spôsob, pomocou ktorého by sa dali priamo prideľovať za sebou nasledujúce úlohy alebo ich inštancie (aj za behu biznis procesu).
	8) Capability-Based Allocation	-	Obe špecifikácie popisujú o zdrojoch len to, akú majú rolu, akú zastávajú úlohu v organizačnej štruktúre, ale žiadne informácie o ich schopnostiach.
	9) History-Based Allocation	-	Obdobne ako pri vzore 8).
	10) Organisational Allocation	-	Je podporované len priradenie úlohy zoznamu rolí alebo zoznamu zdrojov. Nie je však možné ju priradiť na základe organizačných vzťahov.
	11) Automatic Execution	X	Každá úloha (či už ľudská úloha, nejaký proces, webová služba, ...) môže byť vykonaná už priamo vďaka aktivite `<invoke>` v štandardnom BPEL bez toho, aby musela byť priradená zdroju.
Push Patterns	12) Distribution by Offer - Single Resource	X	Pomocou konštrukcie *notification* sa v jej atribúte *documentation* dá poslať ľubovoľná informácia – teda aj o úlohách. Je už len na zdroji, či úlohu prevezme a čo s ňou ďalej spraví. Notifikácia podporuje adresovanie zoznamu zdrojov alebo skupín.
	13) Distribution by Offer - Multiple Resources	X	Rovnako ako pri vzore 12)
	14) Distribution by Allocation - Single Resource	X	Vykonávacie prostredie pre biznis proces vytvorí úlohu a pri priraďovaní úlohy zdroju zaručí, že úlohu dostane práve jeden zdroj.
	15) Random Allocation	-	Vykonávacie prostredie pre biznis proces pri prideľovaní zdrojov úlohe môže využiť ľubovoľnú *expressionLanguage* na vytvorenie dotazov. Štandardne sú však podporované len niektoré funkcie jazyka XPath, takže tento vzor nie je podporovaný priamo.
	16) Round Robin Allocation	-	Rovnako ako pri vzore 15).
	17) Shortest Queue	-	Rovnako ako pri vzore 15).
	18) Early	*	Pomocou notifikácií možno poslať ľubovoľné

skupina vzorov	vzor	podpora	poznámky
	Distribution		informácie a teda aj o prípadných budúcich úlohách. Nie je však možné priradiť takúto úlohu zdroju.
	19) Distribution on Enablement	X	Tento vzor popisuje štandardné prideľovanie úloh zdrojom popísané vo WS-HT.
	20) Late Distribution	-	Vzor nie je podporovaný, lebo podľa WS-HT 4.7.1 Normal Processing of a Human Task je tretím bodom pri vytvorení úlohy pridelenie roly alebo zoznamu zdrojov k nej.
Pull patterns	21) Resource-Initiated Allocation	X	Vzor je podporovaný pomocou operácií „claim“ a „start“ popísaných vo WS-HT – časť 6.1.1 Participant Operations. V definícii operácie „start“ sa nehovorí, kedy musí byť vykonaná. Teda nemusí to byť ihneď po operácii „claim“.
	22) Resource-Initiated Execution - Allocated Work Item	X	Pomocou operácie „claim“ popísanej vo WS-HT – časť 6.1.1 Participant Operations.
	23) Resource-Initiated Execution - Offered Work Item	X	Pomocou operácií „claim“ a „start“ popísaných vo WS-HT – časť 6.1.1 Participant Operations. V definícii operácie start sa nehovorí, kedy musí byť vykonaná. Teda môže to byť ihneď po operácii „claim“.
	24) System-Determined Work Queue Content	-	Ani jedna zo špecifikácií neposkytuje možnosť vynútenia si nejakého usporiadania zoznamu úloh zdroja na základe ich obsahu alebo vlastností.
	25) Resource-Determined Work Queue Content	-	Zdroj si nemôže nijako upravovať svoj zoznam úloh. Môže si z neho iba vybrať úlohu na základe určitých jeho atribútov pomocou atribútu `searchBy` v elemente `task`.
	26) Selection Autonomy	X	Pomocou operácie „claim“ popísanej vo WS-HT – časť 6.1.1 Participant Operations a operácií popísaných vo WS-HT – v časti 6.2 XPath Extension Functions (najmä vďaka funkcii getInput, ktorá vracia vstupnú správu spojenú s úlohou – tu teda môžu byť zapísané niektoré vlastnosti úlohy).
Detour Patterns	27) Delegation	X	Pozri WS-HT 4.7.3 Delegating or Forwarding a Human Task.
	28) Escalation	X	Pozri časť 4.6 vo WS-HT.
	29) Deallocation	X	Pomocou operácie „release“ popísanej vo WS-HT – časť 6.1.1 Participant Operations.
	30) Stateful Reallocation	X	Pomocou postupnosti operácií “release” a “delegate” popísaných vo WS-HT – časť 6.1.1 Participant Operations

skupina vzorov	vzor	podpora	poznámky
	31) Stateless Reallocation	X	Časť 4.7.4 vo WS-HT – v nej sa nič nehovorí o tom, či pozastavená úloha zostáva pri zdroji, ktorý ho pozastavil. Preto úlohu môžu vykonať aj iné zdroje.
	32) Suspension / Resumption	X	Pozri „actual owner“, časť 4.7.4 vo WS-HT – v nej sa nič nehovorí o tom, či pozastavená úloha zostáva pri zdroji, ktorý ho pozastavil. Teda vo všeobecnosti túto pozastavenú úlohu môžu vykonať aj iné zdroje.
	33) Skip	X	Pozri časť 4.7.5 vo WS-HT a funkciu „skip“ v časti 6.1.1 Participant Operations.
	34) Redo	-	Ani jedna zo špecifikácií neposkytuje možnosť distribúcie alebo vykonania úlohy, ktorá už raz bola vykonaná.
	35) Pre-Do	-	Ani jedna zo špecifikácií neumožňuje za behu biznis procesu meniť poradie pridelenia úloh zdrojom.
Auto-start Patterns	36) Commencement on Creation	-	Zdroj musí zavolať funkciu start na úlohe, takže nie je možné, aby začala úloha ihneď po jej vytvorení.
	37) Commencement on Allocation	-	Rovnako ako predošlý vzor 36).
	38) Piled Execution	-	Ani jedna zo špecifikácií neposkytuje možnosť pre automatické spustenie úloh paralelne s ich distribúciou.
	39) Chained Execution	-	Ani jedna zo špecifikácií neposkytuje možnosť pre automatické spustenie nasledovných úloh, keď už aktuálna úloha bola vykonaná.
Visibility Patterns	40) Configurable Unallocated Work Item Visibility	-	Viditeľnosť úloh sa dá nastaviť len v čase návrhu biznis procesu – podľa toho, kde v kóde (t. j. do ktorého rozsahu viditeľnosti) aktivity `<peopleActivity>` dáme WS humanTask (pozri BPEL4people – časť 4 People Activity). Viditeľnosť nie je možné meniť na základe toho, v akom stave sa úloha práve nachádza.
	41) Configurable Allocated Work Item Visibility	-	Rovnako ako predošlý vzor 40).
Multiple Resource Patterns	42) Simultaneous Execution	X	Keďže humanTask je súčasťou aktivity *people activity* (definovanej v BPEL4people časť 4 People Activity) a tá môže byť súčasťou ľubovoľnej aktivity v BPEL (teda aj súčasťou *flow* alebo paralelného *foreach* cyklu) a zároveň, ak je definovaný rovnaký vlastník úlohy (buď ako literál alebo ako výsledok dotazu), tak je možné, aby jeden zdroj (človek)

skupina vzorov	vzor	podpora	poznámky
			vykonával paralelne viacero úloh.
	43) Additional Resources	-	Ani jedna zo špecifikácií BPEL4people ani WSHT neponúka rozdelenie úlohy medzi viacero zdrojov.

Tabuľka 3.1: BPEL4People a jeho podpora pre workflow resource patterns.

Vysvetlivky: X – úplná podpora daného vzoru; * čiastočná podpora daného vzoru; - nepodporovaný vzor; WS-HT je skratka pre dokument [12], BPEL4people je skratka pre dokument [13]; Jednotlivé vzory sú popísané v časti 4.5, str. 43.

3.2 Iné doteraz vykonané práce

K oblasti, ktorou sa zaoberáme v našej diplomovej práci, sme našli tri relevantné články. Ich hlavné myšlienky sú obsahom nasledovných častí. V časti 4.6, Prínosy nášho systému, na str. 54 porovnávame tieto práce s naším prístupom.

3.2.1 Systém VieBOP

VieBOP [14] je externý systém (vzhľadom na ľubovoľné vykonávacie prostredie BPEL) skladajúci sa z viacerých komponentov. Jeho hlavnou snahou je umožniť vykonávanie BPEL kódu obsahujúceho konštrukcie z rozšírenia BPEL4People a súvisiacej špecifikácie Web Services Human Task bez toho, aby vykonávacie prostredie BPEL muselo toto rozšírenie podporovať. Podľa autorov by mal systém VieBOP spolupracovať s každým BPEL vykonávacím prostredím, ktoré spĺňa špecifikáciu jazyka BPEL 2.0.

Systém VieBOP funguje takto:

1 Ak vykonávacie prostredie BPEL narazí pri interpretovaní BPEL zdrojového textu na ľudské úlohy popísané pomocou rozšírenia BPEL4People, tak ich popis je poslaný do systému VieBOP.

2 Systém VieBOP ľudské úlohy preloží do štandardného BPEL kódu[7], v ktorom sa nepoužívajú žiadne konštrukcie z rozšírenia BPEL4People. Potom vykonávanie biznis procesu pokračuje takto (popíšeme iba postup pre preklad a vykonanie `people activity`):

2.1 VieBOP si vytvorí svoju internú reprezentáciu biznis procesu obsahujúcu aj ľudské úlohy.

2.2 Každá `people activity` je transformovaná na dvojicu aktivít: `invoke` a `receive` zo štandardného BPEL.

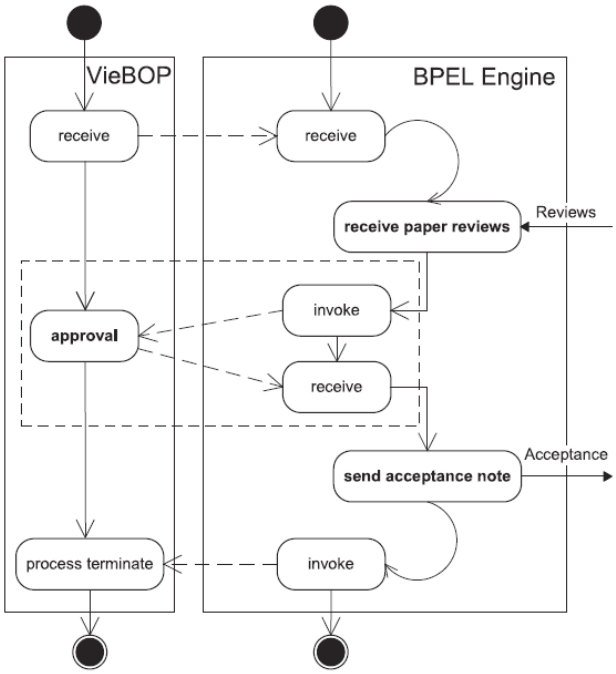
2.3 Vygenerovaná aktivita `invoke` (z BPEL vykonávacieho prostredia) zavolá `people activity` (cez volanie webovej služby poskytovanej systémom VieBOP) nachádzajúcu sa na strane VieBOP.

2.4 Po dokončení `people activity` na strane VieBOP, VieBOP zavolá vygenerovanú aktivitu `receive` na strane vykonávacieho prostredia BPEL.

2.5 Keď je proces na strane vykonávacieho prostredia BPEL tesne pred svojím ukončením, ešte o tom pošle informáciu systému VieBOP pomocou `invoke` aktivity a VieBOP ukončí svoju internú reprezentáciu biznis procesu.

7 Žiaľ v článku sa nepíše, či sa konštrukcie z BPEL4People preložia pred vykonávaním biznis procesu, alebo až za jeho behu, a ako sa dosahuje, že vykonávacie prostredie BPEL vykonáva preložený a nie pôvodný kód.

3 Okrem prekladu z BPEL4People do štandardného BPEL si VieBOP udržuje aj



Obrázok 3.1: Preklad people activity do štandardného BPEL kódu a jeho vykonanie pomocou systému VieBOP. Obrázok je prevzatý z práce [14], str. 549.

aktuálne stavy úloh a ďalšie informácie o nich.

Popísaný proces je zobrazený aj na obr 3.1.

Podstatným komponentom systému VieBOP je *HR Server* spravujúci informácie špecifické pre BPEL4People. Tento komponent je možné nahradiť ľubovoľným iným komponentom (napr. z dôvodu prispôsobenia systému vlastným potrebám).

3.2.2 XML interaktívna služba pre biznis proces aplikácie

Autori tejto práce [15] vytvorili systém nazvaný **XML interaktívna webová služba (XMLIS)**. Služba má ambíciu byť všeobecne použiteľná pre ľudské úlohy, a preto je nezávislá od prostredia vykonávajúceho biznis procesy. Človek interaguje s XMLIS pomocou bežných HTML formulárov. XMLIS sa obmedzuje iba na funkcie úzko súvisiace s ich generovaním a zobrazovaním a neposkytuje žiadne funkcie, ktoré neslúžia na tento účel.

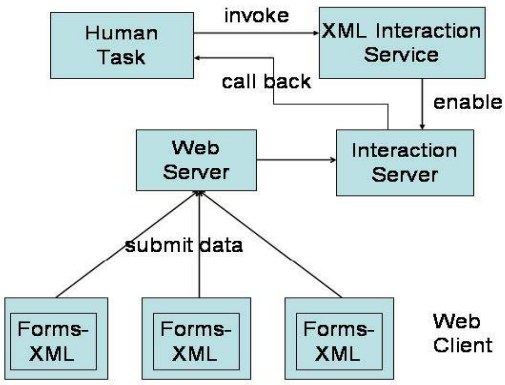
Systém teda z konceptuálneho hľadiska poskytuje iba platformu pre generovanie formulárov, no nezabezpečuje dodržiavanie takmer žiadnych vlastností týkajúcich sa práce s ľuďmi (napríklad niektoré z workflow patterns).

3.2.2.1 Princíp fungovania XML interaktívnej služby a jej komponenty

Na obrázku 3.2 je zobrazená architektúra služby XMLIS. Ľudská úloha spúšťa XMLIS tak, že vytvorí požiadavku. Na uspokojenie požiadavky XMLIS spustí komponent nazvaný **Interakčný server (IS)** bežiaci na strane web servera. IS je komponent, ktorý generuje z XML súborov HTML formuláre a odosiela ich vo forme web stránok používateľovi. Okrem generovania umožňuje

používateľovi vykonať len nasledovné operácie: *submit, complete, OK* a *abort*.

Samotné generovanie HTML formulárov z XML súborov spĺňajúcich schému, ktorá je pre ne definovaná, zabezpečuje komponent **Forms-XML**. Forms-XML podporuje tiež vytváranie viacstránkových formulárov a autori preň naprogramovali aj grafický editor určený pre neprogramátorov.



Obrázok 3.2: Architektúra služby XMLIS.
Obrázok je prevzatý z práce [15].

3.2.3 Pervasive Enablement of Business Processes

Je podstatné poznamenať, že tento článok [16] bol publikovaný ešte pred vydaním konceptu (*whitepaper*) špecifikácie BPEL4People.

Autori tejto práce vytvorili systém integrujúci technológie pre zapojenie ľudí do biznis procesov a podporu ich spolupráce pomocou rôznych komunikačných prostriedkov (napr. e-mail, sms, instant messaging). Pre podporu ľudských úloh definovali syntaktické rozšírenie jazyka BPEL.

3.2.3.1 Hlavné myšlienky a prínosy článku

1 Autori **pridali do syntaxe BPEL tieto konštrukcie pre podporu ľudských úloh**:

1.1 **`<Human Partner>`** definuje ľudskú entitu ako účastníka biznis procesu.

1.2 Konštrukciu **`<interact>`** typu ***Process-to-People***. Používa sa na modelovanie komunikácie medzi `<human partners>` a BPEL procesom.

1.3 Konštrukcie typu ***People-to-People***. Používajú sa na modelovanie priamej komunikácie medzi ľudskými účastníkmi biznis procesu. Sú to konštrukcie **`<notify>`** a **`<converse>`** (ako odpoveď ku `<notify>`).

2 Autori prinášajú **možnosť zapojenia ľudí do biznis procesov pomocou rôznych komunikačných prostriedkov** (nielen pomocou webových formulárov) a ich konkrétny výber podľa aktuálnych okolností a preferencií používateľa (napr. spolupráca cez mail, instant messaging, e-mail, sms, …)

Príklad: *Nadriadený* má potvrdiť pracovnú cestu *zamestnanca*, ale má aktuálne prístup len k mobilnému telefónu. Preto mu prídu potrebné informácie vo forme sms správ a pomocou sms môže vyplniť aj zodpovedajúci formulár pre potvrdenie, resp. zamietnutie cesty.

3.2.3.2 Komponenty systému a ich funkcie

1 Interaction controller (IC)

1.1 Funguje ako proxy pre všetky ľudské úlohy a spravuje o nich informácie. Je realizovaný ako webová služba, vďaka čomu je jednoduché volať ho zo štandardného vykonávacieho prostredia BPEL.

1.2 Po príchode požiadavky na IC, IC zistí od komponentu Context Service najvhodnejší spôsob aktuálneho zapojenia účastníka do biznis procesu.

2 Context Service

2.1 Zistí najvhodnejší spôsob aktuálneho zapojenia účastníka do biznis procesu.

2.1 Každý spôsob interakcie (cez mail, sms, ..) je realizovaný jednou tzv. modalitou. V systéme je štandardne implementovaných viacero bežne používaných modalít. V prípade potreby však možno do systému pridať aj ďalšie modality.

4 Návrh a popis nášho riešenia

Ako sme už popísali v predhovore práce, na Univerzite Komenského v Bratislave vznikla potreba zefektívniť vykonávanie často sa opakujúcich úloh (zväčša ide o schvaľovanie rôznych žiadostí). Jednotlivé potreby kladené na riešenie sú popísané v nasledovnej špecifikácii požiadaviek.

4.1 Špecifikácia požadovanej funkčnosti riešenia

Naše riešenie by malo spĺňať nasledovné požiadavky:

1. Malo by rozširovať vykonávacie prostredie pre biznis procesy využívajúce jazyk BPEL o podporu ľudských úloh na základe nejakého široko akceptovaného štandardu – napr. workflow resource patterns alebo iných vedeckých výsledkov.
2. Vzory z workflow resource patterns, ktoré chceme podporovať:

skupina vzorov	vzor	požadovaná podpora naším systémom
Creation patterns	1) Direct Allocation	X
	2) Role-Based Allocation	X
	3) Deferred Allocation	-
	4) Authorisation	-
	5) Separation of Duties	X
	6) Case Handling	X
	7) Retain Familiar	X
	8) Capability-Based Allocation	X
	9) History-Based Allocation	-
	10) Organisational Allocation	-
	11) Automatic Execution	-

skupina vzorov	vzor	požadovaná podpora naším systémom
Push Patterns	12) Distribution by Offer - Single Resource	X
	13) Distribution by Offer - Multiple Resources	X
	14) Distribution by Allocation - Single Resource	X
	15) Random Allocation	X
	16) Round Robin Allocation	X
	17) Shortest Queue	X
	18) Early Distribution	-
	19) Distribution on Enablement	X
	20) Late Distribution	-
Pull patterns	21) Resource-Initiated Allocation	X
	22) Resource-Initiated Execution - Allocated Work Item	X
	23) Resource-Initiated Execution - Offered Work Item	X
	24) System-Determined Work Queue Content	X
	25) Resource-Determined Work Queue Content	X
	26) Selection Autonomy	-
Detour Patterns	27) Delegation	X
	28) Escalation	X
	29) Deallocation	X
	30) Stateful Reallocation	X
	31) Stateless Reallocation	X
	32) Suspension/ Resumption	X
	33) Skip	X
	34) Redo	-
	35) Pre-Do	-
Auto-start Patterns	36) Commencement on Creation	-
	37) Commencement on Allocation	-
	38) Piled Execution	-
	39) Chained Execution	-
Visibility Patterns	40) Configurable Unallocated Work Item Visibility	-
	41) Configurable Allocated Work Item Visibility	-
Multiple Resource Patterns	42) Simultaneous Execution	X
	43) Additional Resources	-

Tabuľka 4.1: Podpora Resource Patterns naším systémom

Vysvetlivky: X – vzor bude určite podporovaný; - vzor nebude podporovaný

3. Naše riešenie by malo byť nezávislé na vykonávacom prostredí jazyka BPEL, aby jeho použiteľnosť bola čo najširšia. Jedinou podmienkou na vykonávacie prostredie je jeho podpora pre verziu 2.0 jazyka BPEL.

4. Malo by byť čo najjednoduchšie použiteľné a nasaditeľné v kombinácii s vykonávacím prostredím jazyka BPEL.

4.2 Sonic BPEL server

Hoci jednou z požiadaviek na náš systém je, aby bol nezávislý na vykonávacom prostredí jazyka BPEL, pre implementáciu je potrebné vybrať nejaký konkrétny produkt. Na Univerzite Komenského je zakúpený softvér Progress Sonic BPEL server, a preto sa využíva aj v našom systéme.

4.3 Už existujúce komponenty

Nasledujú systémy, ktoré sú vytvorené z komponentov a uvažovali sme o ich použití (resp. niektorých ich komponentov) pri analýze a návrhu nášho systému.

- **YAWL systém** (http://yawlfoundation.org/) – *Worklist Web Application Component.* Je ťažké povedať, do akej miery je tento komponent použiteľný, keďže podporuje len biznis proces jazyk YAWL a nie BPEL. Podporuje však aj väčšinu vzorov z workflow resource patterns, takže pre prácu s ľudskými úlohami je vo všeobecnosti vhodný.

- **Open source WorkFlow Environment** (http://www.openwfe.org/). Ponúka komponenty použiteľné pre spravovanie úloh a aj pre interakciu s nimi pomocou webovej aplikácie, avšak tento projekt je slabo zdokumentovaný a nedávno sa začal vyvíjať v jazyku ruby. To by znamenalo potrebu si ho naštudovať, čo by bolo časovo náročnejšie ako vybrať iný produkt, alebo si vytvoriť komponenty sám.

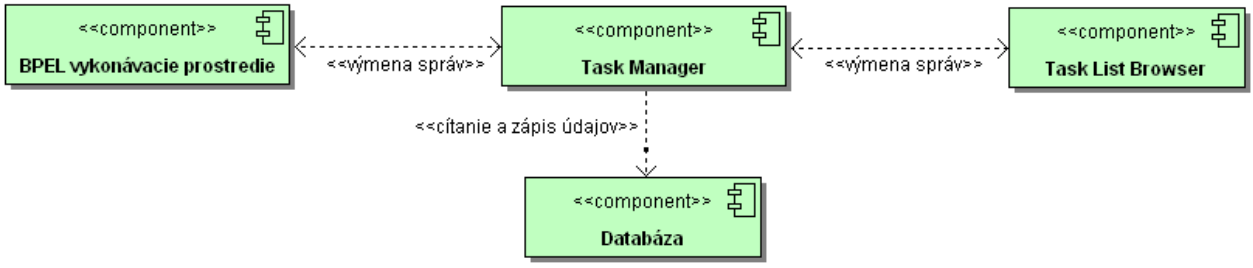
- **Windows workflow foundation** (http://msdn2.microsoft.com/en-us/netframework/aa663328.aspx) – je súčasťou platformy .NET od verzie 3.0. Tento balík však nie je použiteľný v našom systéme, lebo neobsahuje žiadnu službu, ktorá by sa aspoň približne zhodovala s nami definovanými komponentami potrebnými pre implementáciu (alebo aspoň s niektorým z nich).

Z uvedených dôvodov sme sa rozhodli, že nevyužijeme pri návrhu a implementácii nášho systému žiaden z doteraz vytvorených systémov ani nijaký ich komponent.

4.4 Architektúra nášho systému

Celý systém pozostáva z nasledovných komponentov: *Business procesu* bežiacemu vďaka vykonávaciemu prostrediu BPEL (v našom konkrétnom prípade ide o produkt Sonic BPEL server), *Task Manager*-a (TM), *Task List Browser*-a (TLB) a *databázy*. Nasleduje podrobný popis komponentov. Statická architektúra systému je na obr. 4.1 vo forme UML komponentového diagramu.

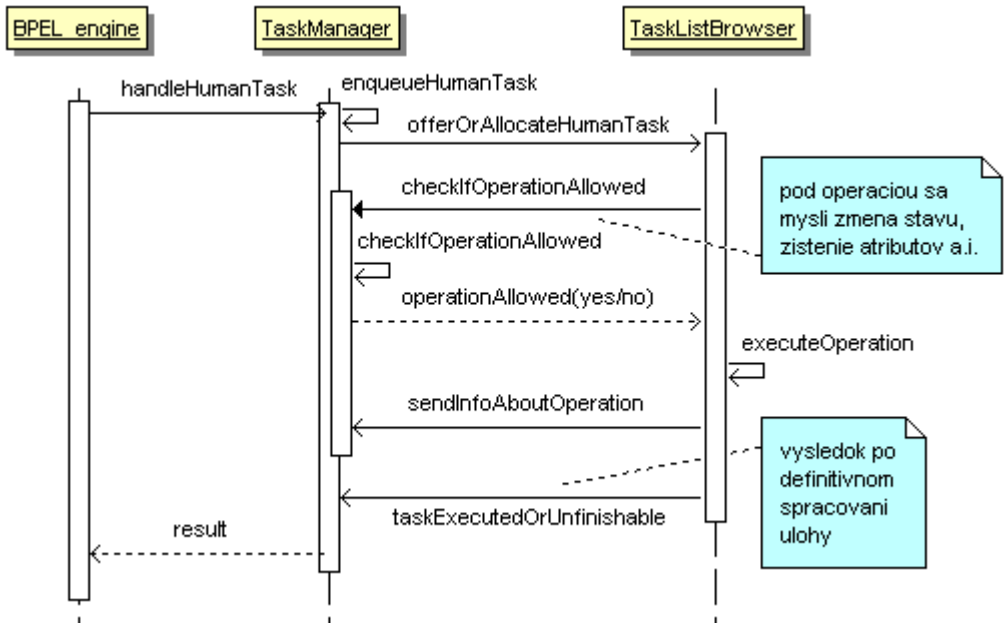
Spracovanie ľudskej úlohy v našom systéme je zobrazené na obr. 4.2 vo forme sekvenčného UML diagramu a prebieha nasledovne:



Obrázok 4.1: Architektúra nášho systému na úrovni komponentov.

1 Vykonávacie prostredie BPEL deleguje požiadavku na spracovanie ľudskej úlohy na TM pomocou BPEL aktivity `<invoke>`. (V diagrame je to zobrazené pomocou operácie handleHumanTask.)

2 TM zaradí úlohu do svojho zoznamu úloh (enqueueHumanTask) a spravuje počas jej životného cyklu jej stav a ostatné vlastnosti. Keď sa nájde zdroj, ktorý môže úlohu prijať, tak mu ju TM pridelí v stave, ktorý je ovplyvnený aplikovanými vzormi



Obrázok 4.2: Spracovanie novej ľudskej úlohy - sekvenčný UML diagram

(offerOrAllocateHumanTask).

3 TLB zobrazuje jednotlivým prihláseným ľuďom zoznamy úloh a ich vlastností. Pred každou operáciou s úlohami, ktorú chce zdroj vykonať (najčastejšie ide o zmenu ich stavu), TLB zistí, či je operácia povolená pomocou delegovania tejto kontroly na TM (checkIfOperationAllowed). TM pošle výsledok operácie checkIfOperationAllowed do TLB. Zdroj potom v prípade pozitívneho výsledku môže (ale nemusí) vykonať danú operáciu (executeOperation) cez webovú stránku TLB. Ak zdroj operáciu vykoná, pošle sa o tom informácia do TM (sendInfoAboutOperation) a TM adekvátne upraví všetky súvisiace atribúty úlohy.

4 Práca s úlohou pokračuje vykonávaním ďalších operácií.

5 Keď úloha prejde do niektorého zo stavov *vykonaná* alebo *nedokončiteľná*, tak TLB o tom pošle informáciu do TM (taskExecutedOrUnfinishable). Následne nato TM vráti odpoveď vykonávaciemu prostrediu BPEL cez rozhranie webovej služby (result) obsahujúcu výsledný stav úlohy spolu s nepovinnými prílohami a nepovinným atribútom *approved*.

V ďalších častiach je popísaná architektúra detailnejšie s podstatnými implementačnými poznámkami na úrovni jednotlivých komponentov.

4.4.1 BPEL Business proces

Každý biznis proces zapísaný v jazyku BPEL môže v našom systéme používať ľudské úlohy nasledovne:

Programátor v BPEL kóde použije štandardnú aktivitu `<invoke>` a nastaví v nej vhodne atribúty tak, aby táto aktivita volala webovú službu poskytovanú TM, ktorá manažuje vykonávanie ľudských úloh. Podstatným pri aktivite `<invoke>` je atribút `inputVariable` typu wsdl message.

```
<invoke inputVariable="taskForTM"
  name="PosliFormularZiadatelovi"
  operation="handleHumanTask"
  outputVariable="responseFromTM"
  partnerLink="humanTasks" xml:ID="12"/>
```

Text 1: Fragment BPEL zdrojového kódu popisujúci volanie našej webovej služby, ktorá manažuje ľudské úlohy.

Atribút `inputVariable` v skutočnosti popisuje dátový typ definovaný pomocou XML schémy. (Jej fragment sa nachádza pod vysvetlením, čo jednotlivé atribúty definujú. Celú XML schému spolu so súvisiacim WSDL súborom môže čitateľ nájsť v prílohe na str. 64.) Tento dátový typ obsahuje tri elementy:

- `WF_patterns` popisuje aplikované vzory na úlohu a s nimi súvisiace atribúty.
- `predefinedHT_Name` popisuje nejakú z preddefinovaných úloh, ktorá má nastavené vzory a s nimi súvisiace vlastnosti.

  Element `<xsd:choice>` zabezpečuje, že z predošlých dvoch atribútov sa vždy môže použiť práve jeden.

- `Data` popisuje ľubovoľné dáta, ktoré budú pridané do posielanej XML správy (napr. formuláre, vysvetľujúce dokumenty, obrázky a pod.). Tieto pripojené dáta sú súčasťou úlohy počas celého jej životného cyklu a môžu byť upravené ľuďmi cez komponent TLB. Ľubovoľné textové dáta sa dajú do XML schémy vložiť ako obsah elementu typu string. Binárne dáta možno pomocou XML schémy uložiť ako typ elementu hexBinary – avšak ich zakódovanie a dekódovanie si programátor musí spraviť sám.

Ak programátor použije atribút `WF_patterns`, má výhodu úplnej kontroly nad aplikovanými vzormi a atribútmi, musí v nich však mať dostatočne podrobný prehľad.

Ak programátor použije atribút `predefinedHT_Name`, tak má obmedzené možnosti, no preddefinované úlohy môže rýchlejšie používať. Okrem toho má možnosť vytvoriť si repozitár vlastných úloh, ktoré môže zdieľať aj s inými spolupracovníkmi.

```
<xsd:element name="handleHumanTask">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:choice>
        <xsd:element name="WF_patterns" type="ht:header"/>
        <xsd:element name="predefinedHT_Name" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="Data" type="ht:Data" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Text 2: Fragment XML schémy popisujúci typy skryté za atribútom inputVariable.

Pre podporu ľudských úloh naším systémom v jazyku BPEL neboli pridané žiadne syntaktické rozšírenia, vďaka čomu nie je potrebné meniť aktuálne používané vykonávacie prostredie BPEL. Samozrejme je nutné použiť nami navrhovaný externý systém.

4.4.2 Task Manager

4.4.2.1 Úlohy

1 Poskytuje bežiacu webovú službu, ktorá prijíma požiadavky na spracovanie úloh pre ľudí a vracia štruktúrovanú odpoveď po ich ukončení pre vykonávacie prostredie BPEL.WSDL súbor spolu so súvisiacou XML schémou je v prílohe na str. 64.

2 Udržiava a vhodne upravuje zoznam úloh aj s ich atribútmi (aplikované vzory, lehoty, a.i.), ktoré prišli z BPEL servera a stavy, v ktorých sa úlohy práve nachádzajú.

3 Zabezpečuje dodržiavanie aplikovaných vzorov na úlohy a zdroje a z nich vyplývajúcich obmedzení. (Napr. kontroluje, či je možné zmeniť stav úlohy pri aplikovaných vzoroch.)

4 Vyhodnocuje dôležitosť vzorov (ak sa nejakej úlohy alebo zdroja týka viacero vzorov) a na ich základe vyplývajúcich obmedzení vyberá tie zdroje, ktoré majú právo, schopnosti, resp. sú najvhodnejší kandidáti, aby vybranú úlohu vykonali, alebo im bola ponúknutá.

4.1 Sem spadá aj prideľovanie úloh konkrétnej množine ľudí na základe rolí, vyžadovaných schopností pre vykonávanie úloh a pod.

5 V prípade potreby (napr. ak pri aktuálnom stave prerozdelenia úloh zdrojom nemožno nejakú úlohu prideliť žiadnemu zdroju) alebo pre zefektívnenie vykonávania úloh (*load balancing*) vhodne mení pridelenie úloh zdrojom (vzory 28) Escalation, 29) Deallocation a 31) Stateless Reallocation). *Load balancing* pre jednoduchosť nebudeme podporovať.

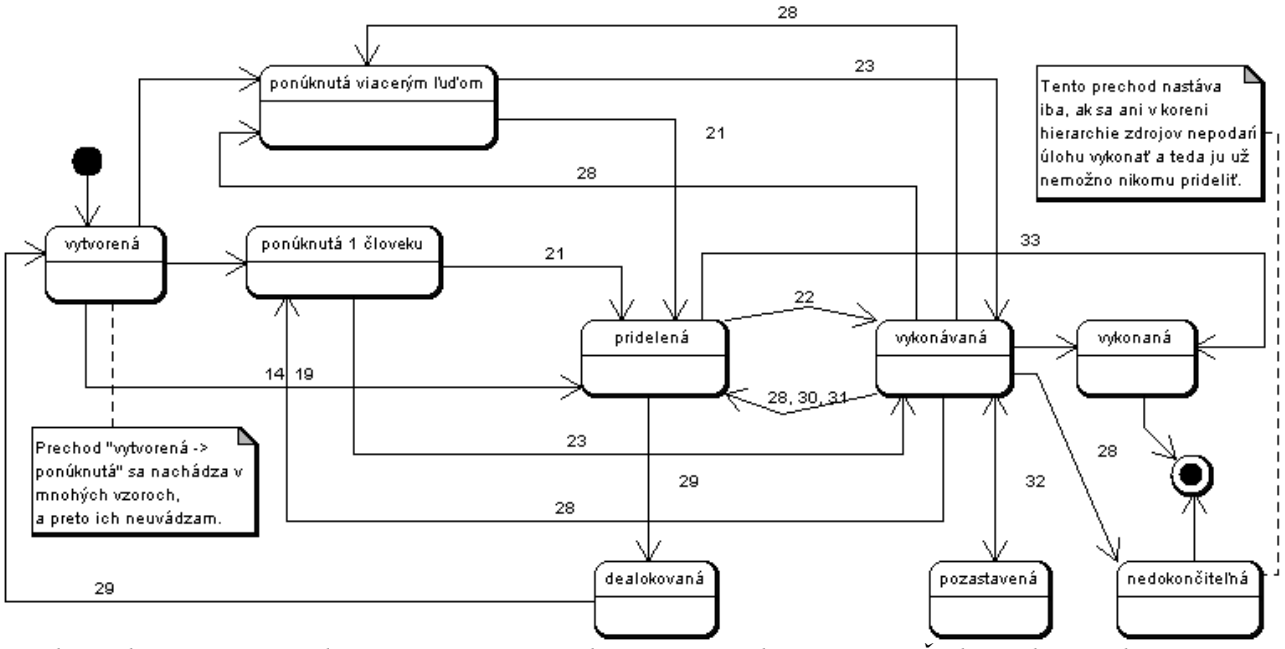
6 Výmena správ s BPEL serverom, s TLB a aktualizovanie informácií o úlohách na základe týchto správ (najmä zmeny stavov úloh a ich presúvanie medzi zdrojmi).

7 Úlohy vyplývajúce z ostatných aplikovaných vzorov sú také, ako je popísané v časti 4.5.1, str. 43.

4.4.2.2 Implementácia

Task Manager predstavuje serverový komponent systému. Implementáciu sme robili v programovacom jazyku java (java SE 1.6). Webová služba je implementovaná s použitím knižníc Java API for XML Web Services 2.0 (JAX-WS), ktoré sú od verzie java SE 1.6 jej štandardnou súčasťou a je popísaná prislúchajúcim WSDL súborom.

Dodržiavanie obmedzení, ktoré vyplývajú z aplikovaných vzorov – najmä prechody medzi jednotlivými stavmi, v ktorých sa úloha môže nachádzať, sme realizovali pomocou využitia dizajnového vzoru „*state pattern*“[17]. Pri väčšine vzorov je popísané, v akom stave sa úloha alebo zdroj, na ktoré je daný vzor aplikovaný, môže nachádzať a aj to, do akých nasledujúcich stavov sa môže dostať. Analýzou týchto popisov sme vytvorili stavový diagram zobrazený na obrázku 4.3.



Obrázok 4.3: Stavový diagram pre vzory vybrané na implementáciu. Čísla na hranách popisujú vzory, ktoré musia byť aplikované na úlohu a / alebo zdroj, aby sa daná zmena stavu mohla uskutočniť.

Zoznam vzorov s ich číslami: 14) Distribution by Allocation - Single Resource, 19) Distribution on Enablement, 21) Resource-Initiated Allocation, 22) Resource-Initiated Execution - Allocated Work Item, 23) Resource-Initiated Execution - Offered Work Item, 28) Escalation, 29) Deallocation, 30) Stateful Reallocation, 31) Stateless Reallocation, 32) Suspension/ Resumption, 33) Skip.

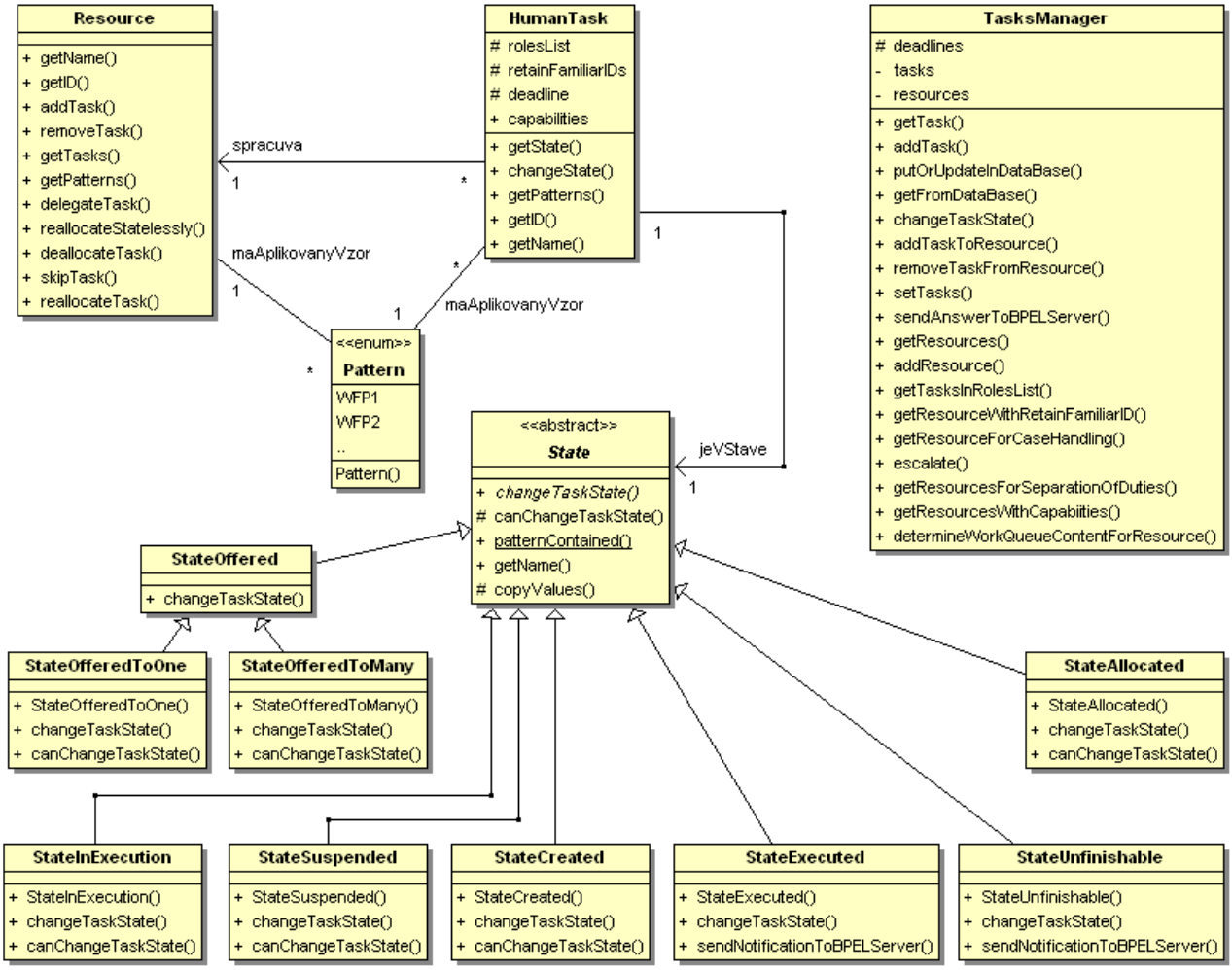
Na obrázku 4.4 je UML diagram tried pre TM zobrazujúci jeho hlavné triedy a ich vzťahy. Trieda **HumanTask** v ňom predstavuje ľudskú úlohu. Vymenované sú jej atribúty potrebné pre podporu vzorov. Pomocou metódy `changeState()` sa mení stav úlohy delegovaním na metódu `State.changeState()` a metóda `getPatterns()` vracia vzory aplikované na úlohu. To, že vzor je aplikovaný na úlohu, znamená, že úloha ho má v svojom zozname vzorov a je potrebné vlastnosti tohto vzoru dodržiavať. Okrem toho to tiež znamená, že vzor môže byť aplikovaný na úlohu, lebo nie všetky vzory sa dajú aplikovať na úlohy – niektoré sa dajú aplikovať len na zdroje, niektoré iba na vykonávacie prostredie BPEL.

Trieda **Resource** predstavuje zdroj. Sú v nej metódy pre podporu vzorov týkajúcich sa zdrojov: `delegateTask()`, `reallocateStatelessly()`, `deallocateTask()`, `skipTask()`, `reallocateTask()`.

Trieda **TasksManager** je samotné jadro komponentu. Okrem dátových štruktúr a metód pre prácu s úlohami a zdrojmi obsahuje aj metódy pre podporu vzorovtýkajúcich sa biznis proces vykonávacieho prostredia: `getTasksInRolesList()`, `getResourceWithRetainFamiliarID()`, `getResourceForCaseHandling()`, `escalate()`, `getResourcesForSeparationOfDuties()`, `getResourcesWithCapabiities()`, `determineWorkQueueContentForResource()`. Metódy

`getRandomTask()`, `getRoundRobinTask()`, `getShortestQueueTask()` sú určené pre podporu vzorov 15 – 17.

Abstraktná trieda **State** a jej podtriedy reprezentujú jednotlivé stavy, v ktorých sa úlohy môžu nachádzať a realizujú už spomínaný stavový diagram využitím dizajnového vzoru *state pattern*. Pred pokusom o zmenu aktuálneho stavu metódou `changeState()` na iný sa vždy zavolá metóda `canChangeTaskState()`, ktorá overuje, či pri daných aplikovaných vzoroch na úlohu a zdroj je možné zmenu vykonať. Podtriedy StateExecuted a StateUnfinishable obsahujú navyše aj metódu `sendNotificationToBPELServer()`, ktorá posiela výsledok o vykonaní úlohy do BPEL servera.



Obrázok 4.4: UML diagram tried pre komponent Task manager

4.4.3 Task list browser

4.4.3.1 Úlohy

1 TLB zobrazuje zoznam úloh s ich stavmi a atribútmi, ktoré prišli z TM.

1.1 Pre úlohy platí (okrem vzoru 42) Simultaneous Execution), že ak je niektorá z úloh zdroja v stave vykonávania, tak zdroj nemôže vykonávať ďalšiu úlohu a teda za tejto podmienky TLB zdroju neumožňuje zmeniť stav na žiadnej inej úlohe na vykonávanú.

1.2 Aby vyhovel TLB jednotlivým vzorom, tak podporuje rôzne spôsoby zobrazenia úloh:

1.2.1 Zdroj si nemôže vyberať úlohu – tá je mu vždy ponúknutá ako jediná možnosť. O ostatných úlohách síce vie, avšak nemôže ovplyvniť ich poradie vo svojom zozname úloh .

1.2.2 Zdroj si môže vybrať jednu z viacerých úloh (napr. ak je ich viacero v stave ponúknutá).

2 Ak úloha obsahuje aj priložené dáta vo forme formulára, tieto sú užívateľovi zobrazené na vyplnenie.

3 Ak úloha obsahuje aj priložené dáta v inej ako formulárovej forme, užívateľ má možnosť ich uložiť a prezrieť si ich na svojom počítači. Taktiež má možnosť odstrániť, alebo pridať nové prílohy. (Ak má na to oprávnenie.)

4 TLB umožňuje prideliť skupinu úloh zdroju (vzory 6) Case Handling a 7) Retain Familiar) a aj ich skupinové delegovanie.

5 O každej operácii vykonanej na úlohe posiela informáciu TM.

4.4.3.2 Implementácia

Task list browser je realizovaný pomocou webovej aplikácie založenej na JSP stránkach. Ak sú k úlohe pripojené aj dáta vo forme formulára (XML súbor spĺňajúci XML schému pre formuláre), tak tento XML formulár je pretransformovaný pomocou XSLT dokumentu na HTML formulár, ktorý môže byť zobrazený každým dnes bežne dostupným webovým prehliadačom. Do úvahy pri ukladaní formulárov a získavaní vyplnených dát z nich prichádzalo viacero možností. Stručne urobíme ich prehľad v ďalšej časti a odôvodníme náš výber.

4.4.3.3 Presun formulárov a používateľova interakcia s nimi

1 Možnosť č. 1 transformácia pomocou XSLT

1.1 Dáta popisujúce formulár vo forme XML poslané z BPEL servera (sú súčasťou popisu úlohy) sú transformované cez XSLT definovaný dizajnérom biznis procesu a zobrazia sa účastníkovi biznis procesu ako webová stránka obsahujúca HTML formulár.

1.2 Používateľ vyplní ponúknutý formulár a klikne na tlačidlo odoslať alebo zrušiť. (Je úlohou dizajnéra biznis procesu, aby XSLT dokument vytvoril tak, že v ňom budú tlačidlá odoslať a zrušiť odkazovať na správne JSP stránky resp. servlety TLB.)

1.3 Príslušná JSP stránka v TLB extrahuje formulárové dáta a pripojí ich k správe vo vhodnej forme, ktorá sa vracia BPEL serveru.

Správa má nasledovnú štruktúru:
wf:Header – údaje pre TM resp. TLB popisujúce aplikované vzory na úlohu a s nimi súvisiace parametre,
wf:Data – „čisté“ aplikačné dáta; napr. nejaký XML formulár, ktorý je ako súčasť procesu potrebné vyplniť (napr. obsahujúci požiadavku na preplatenie cestovného počas pracovnej cesty),
wf:UserResponse – tu sa ukladá odpoveď používateľa.

- **Výhody:**
  - Riešenie je nezávislé na platforme.
  - Nie sú potrebné poplatky za softvérové licencie.
  - Sú využité firemne nezávislé štandardy.

- Je umožnená maximálna konfigurovateľnosť.

- **Nevýhody:**
  - Celú transformáciu je potrebné spraviť vo vlastnej réžii. Z toho vyplýva väčšia časová náročnosť pri vývoji systému.

2 Možnosť č. 2 Microsoft Office InfoPath

2.1 Po výbere úlohy používateľom sa spustí desktopová aplikácia **Microsoft Office InfoPath** s dátami popísanými vo wf:Data (príslušný formulár predpripraví dizajnér biznis proces-u), ktoré InfoPath pretransformuje na formulár.

2.2 Používateľ vyplní formulár a klikne na tlačidlo odoslať, pričom dáta sa spracujú ako v predchádzajúcom prípade.

- **Výhody:**
  - Nie je potrebné programovať časť systému zobrazujúcu formuláre.
  - Je podporované vypĺňanie formulárov, ktoré sú vo forme webovej služby (definovaných pomocou wsdl aj SOAP).
- **Nevýhody:**
  - Sú potrebné poplatky za softvérové licencie.
  - Platforma je obmedzená len na OS Microsoft Windows XP a vyšší.
  - Používa vlastný komerčný XML formát. Tým môže vzniknúť problém s kompatibilitou a portabilitou.
  - Spustenie desktopovej aplikácie volanej z webového prehliadača na klientskom počítači je štandardne z bezpečnostných dôvodov zakázané, nie však nemožné.

3 Možnosť č. 3 XForms

XForms je štandard W3C konzorcia [18]. Jeho idea je, že by mal nahradiť klasické formuláre v HTML stránkach XML formulármi. V svojej internej architektúre využíva osvedčený vzor Model-view-controller (MVC).Vďaka MVC je možné na dáta a ich vzťahy klásť komplexné obmedzenia. Tiež poskytuje možnosti nastavenia preddefinovaných hodnôt a validáciu údajov zohľadňujúcu obmedzenia.

Keďže je to pomerne nový štandard, tak zatiaľ žiaden z dominantných webových prehliadačov nepodporuje natívnu prácu s ním. Pre jeho podporu však existuje viacero rozšírení, resp. zásuvných modulov. Mal by byť tiež súčasťou pripravovaného štandardu XHTML 2.0, a teda je predpoklad, že bude všeobecne podporovaný.

Primárne je tento štandard určený pre webové formuláre, je ale vhodný aj pre desktopové aplikácie na vypĺňanie formulárov, resp. pre ľubovoľné iné formuláre. XForms formulár možno navrhnúť (okrem ručného písania XML) aj pomocou viacerých nástrojov. Mnohé z nich sú komerčné. Existujú však aj možnosti, ktoré sú zadarmo, alebo majú otvorený zdrojový kód – napríklad balík openOffice.org od verzie 2.0 podporuje ich WYSIWYG vytváranie. V prípade openOffice.org je používateľský postup obdobný ako pri aplikácii MS Infopath.

- **Výhody:**
  - Nie je potrebné programovať časť systému zobrazujúcu formuláre.

- Nie sú potrebné poplatky za licencie za software.
- Je využitý firemne nezávislý štandard XForms.
- Existujú platformovo nezávislé riešenia zadarmo alebo s otvoreným kódom (napr. Openoffice.org alebo FormFaces http://www.formfaces.com/) pre podporu vytvárania a vypĺňania XForms.

○ **Nevýhody:**

- Štandard XForms zatiaľ nie je široko používaný.

4 Možnosť č. 4 PDF Forms a XML Forms

Toto sú riešenia vyvinuté firmou Adobe [19]. V prípade PDF Forms možno ku pdf súboru definovať FDF súbor (Forms Data Format – je to derivát pdf súboru), ktorý popisuje jednotlivé polia a iné prvky formulárov. Existujú nástroje na ich ručnú tvorbu (napr. Adobe Acrobat). Adobe poskytuje aj súbor knižníc na spracovávanie údajov zo zaslaných FDF súborov (FDF Toolkit) napr. pre databázové spracovanie.

Novšou obdobou PDF Forms sú XML Forms, kde sú dizajn, validácie a samotné dáta z formulára definované vo forme XML súboru.

Postup pri vykonávaní úlohy pomocou PDF Forms resp. XML Forms je nasledovný:

4.1 Užívateľ vyplní pdf formulár.

4.2 Užívateľ odošle formulár jednou z viacerých poskytovaných možností.

○ **Výhody:**

- Nie je potrebné programovať časť systému zobrazujúcu formuláre.
- Existujú platformovo nezávislé riešenia pre vypĺňanie pdf formulárov, ktoré sú k dispozícii zadarmo. (napr. Adobe acrobat reader, Evince)

○ **Nevýhody:**

- PDF Forms ani XML Forms nie je štandard.
- Je relatívne málo používaný.
- Spustenie desktopovej aplikácie volanej z prehliadača na klientskom počítači je štandardne z bezpečnostných dôvodov zakázané, nie však nemožné.
- Sú potrebné poplatky za licencie za software na tvorbu formulárov.
- Neexistuje spôsob automatického (programového generovania) formulárov.

Z predstavených riešení sa nám ako najlepšie použiteľná zdá v aktuálnom čase možnosť č.1 pre širokú podporu HTML formulárov a ich väčšinou dostatočné vlastnosti. V budúcnosti by bolo vhodné porozmýšľať nad použitím XForms (najmä, ak budú natívne podporované prehliadačmi), keďže umožňujú definovať komplikovanejšie vzťahy aj ich validáciu medzi poliami formulárov bez nutnosti programovania.

4.4.3.4 Inicializácia procesu užívateľom

Inicializácia procesu užívateľom je realizovaná nasledovne:

1 Súčasťou systému je špeciálna web stránka obsahujúca zoznam procesov (vo forme odkazov), ktoré daný užívateľ na základe svojich práv, resp. rolí môže iniciovať.

2 Zo zoznamu procesov si užívateľ vyberie proces, ktorý chce iniciovať. **Kliknutie na linku vybraného procesu** v pozadí skrýva:

2.1 Poslanie požiadavky na JSP stránku resp. servlet o spustenie procesu.

2.2 JSP stránka, resp. servlet pomocou webovej služby spustí inštanciu príslušného procesu na BPEL serveri.

3 Vykonávanie procesu pokračuje ďalej tak, ako je to popísané v časti 4.4.3.3, str. 40.

4.5 Výber vzorov

V nasledovných častiach popíšeme a odôvodníme výber vzorov zo skupiny workflow resource patterns, ktoré zahrnieme, resp. vylúčime zo systému a načrtneme pri každom vzore myšlienku implementácie rozdelenú medzi jednotlivé komponenty nášho systému. Komponenty sú popísané v časti 4.4, str. 34.

V zápise BPEL procesu bude pri všetkých vzoroch uložená aj informácia, aké vzory majú byť aplikované na danú úlohu pomocou atribútu `inputVariable` aktivity `<invoke>`.

Popis jednotlivých vzorov je prebraný z [8] a skrátený tak, aby bolo možné pochopiť, čo jednotlivé vzory vyjadrujú. Viaceré zo zdrojov sa líšia na prvý pohľad len v malých rozdieloch. Názvy vzorov sú ponechané v angličtine, aby si čitateľ mohol jednoduchšie nájsť ich podrobnejší popis.

Pri implementačných poznámkach sú pri každom vzore uvedené len tie komponenty, ktoré na dosiahnutie konkrétneho vzoru poskytujú netriviálnu funkčnosť, alebo sú uvedené pre ucelené pochopenie implementácie. Ak je niektorý zo vzorov aplikovaný na úlohu alebo zdroj, znamená to, že úloha, resp. zdroj musia spĺňať vlastnosti, ktoré vzor popisuje.

4.5.1 Určite podporované vzory

Vzor 1) Direct Distribution

Popis vzoru: Možnosť špecifikovať v čase návrhu biznis procesu identitu zdroja, ktorý vykoná ľudskú úlohu.

Dôvod výberu: Tento vzor patrí medzi základné vzory, a teda je aj často používaný.

Funkčnosť: Ak sa pre niektorý z komponentov pre daný vzor nevyžaduje žiadna funkčnosť, tak tento komponent pri vzore neuvádzam.

Dátové atribúty: Identifikátor zdroja, ktorý má úlohu vykonať.

BPEL server: V zápise BPEL procesu pri danej úlohe musí byť identifikátor konkrétneho zdroja. Informácia o zdroji sa posiela ako všetky ostatné informácie z BPEL servera – teda pomocou konštrukcie `inputVariable`.

TM: Po príchode do systému je úloha štandardne v stave vytvorená. Keď už zdroj definovaný identifikátorom môže prijať ďalšiu úlohu, tak TM mu prišlú úlohu pridelí v stave pridelená.[8] Ak neexistuje zdroj, ktorý je určený identifikátorom v posielanej úlohe, tak sa úloha pridelí ľubovoľnému zdroju, ktorý ju môže vykonať.

8 Keďže v [8] sa pri popise vzorov 1 ani 2 nepíše, v akom stave majú byť zdroju prideľované, ja budem predpokladať, že je to v stave pridelená. V prípade, že bude na úlohu zároveň aplikovaný aj niektorý zo vzorov Distribution by Offer – Single Resource, Distribution by Offer – Multiple Resources alebo Distribution by Allocation – Single Resource, tak tieto vzory majú prednosť.

Vzor 2) Role-Based Distribution

Popis vzoru: Možnosť špecifikovať ***v čase návrhu*** biznis procesu jednu alebo viacero rolí, ktorým budú prideľované inštancie ľudskej úlohy. Inštancie tejto úlohy sú ponúknuté za behu všetkým zdrojom, ktoré majú niektorú z daných rolí.

Dôvod výberu: Tento vzor patrí medzi základné vzory, a teda je aj často používaný.

Funkčnosť: Identifikátorom každej roly je jej meno a rola je reprezentovaná ako zoznam identifikátorov zdrojov. Roly budú uložené v DB.

Dátové atribúty: Pole mien rolí.

BPEL server: V zápise BPEL procesu pri danej úlohe musí byť definovaný neprázdny zoznam rolí, inak sa tento vzor ignoruje.

TM: Ponúkne úlohu každému zdroju, ktorý je v aspoň jednej z uvedených rolí. Ak z TLB dostane správu, že nejaký zdroj začal úlohu vykonávať, vymaže túto úlohu zo zoznamov ponúknutých úloh ostatných zdrojov.

TLB: Zobrazí zdroju úlohu v stave *ponúknutá*[9]. Ak ju nejaký zdroj začne vykonávať, tak TLB pošle o tom TM správu.

Vzor 5) (Separation of Duties)

Popis vzoru: Schopnosť určiť, že vo vykonávanej inštancii biznis procesu musia byť dve úlohy pridelené rôznym zdrojom.

Dôvod výberu: Tento vzor sa najčastejšie používa v prípade, ak je potrebné zaručiť, aby sme dostali dva nezávislé výsledky (napr. názor dvoch rôznych odborníkov na problém). Okrem toho len tri nástroje z porovnávaných na stránke [8] podporujú tento vzor.

Všeobecné implementačné poznámky: Každá úloha z jednej skupiny, zachovávajúca tento vzor, má rovnaké ID. Keďže konkrétna úloha môže byť vykonaná viackrát v jednom biznis procese (napr. v cykle), je potrebné zaručiť, že z množiny zdrojov, ktoré ju budú môcť ďalej vykonať, budú vylúčené všetky zdroje, ktoré ju už doteraz vykonávali.

BPEL server: V zápise správy o danej úlohe musí byť ID skupiny.

TM: Pre každú skupinu úloh sa postupne buduje zoznam zdrojov, ktorým už nemožno prideliť zvyšné úlohy zo skupiny. V prípade, ak niektorá úloha bola pridelená, ale potom nebola vykonaná, tak tento zdroj pridáme opäť do tejto skupiny.

Dátové atribúty: ID skupiny s týmto vzorom.

Vzor 7) (Retain Familiar)

Popis vzoru: V prípade, že je k dispozícii viacero zdrojov, ktoré by sa mohli ujať úlohy, tak pri aplikovaní tohto vzoru je vybraný len ten zdroj, ktorý vykonával aj *predošlú* inštanciu úlohy. Tento vzor minimalizuje časové straty, lebo človek si nemusí zvykať na nový typ úlohy, ani meniť svoje pracovné prostredie. (Napr. iná úloha môže vyžadovať rozdielne informácie ako predošlá a ich získanie trvá nejaký čas.)

Dôvod výberu: Tento vzor je celkom praktický a podporuje ho väčšina z nástrojov porovnávaných v [8].

9 Stav *ponúknutá úloha* má 2 podstavy: ponúknutá 1 človeku a ponúknutá viacerým ľuďom. Toto platí aj pre ďalšie výskyty tohto stavu.

Funkčnosť: Každá skupina úloh, ktorá má tento vzor spĺňať, bude mať svoje ID, lebo takýchto skupín môže byť viacero.

Dátové atribúty: ID skupiny, do ktorej úloha patrí.

TM: Pri prvom výskyte úlohy s daným ID skupiny ponúkne úlohu všetkým zdrojom. Pri ďalších výskytoch úloh s daným ID skupiny úlohy prideľuje len tomu zdroju, ktorý už má dané ID skupiny. Keď je vykonávaná posledná úloha v celej inštancii biznis procesu, tak TM zmaže pri všetkých zdrojoch identifikátory úloh, kvôli šetreniu pamäťou a pre ich možné opätovné použitie.

TLB: Ak nejaký zdroj začne úlohu vykonávať, tak pošle o tom informáciu TM. TM si poznačí, že skupinu úloh s daným ID má ďalej ponúkať len tomuto zdroju.

Vzor 6) (Case Handling)

Popis vzoru: Možnosť prideliť *všetky* úlohy, ktoré sa vyskytujú v rámci jednej inštancie biznis procesu *práve jednému zdroju*.

Dôvod výberu: Tento vzor používa len zoslabené podmienky vzoru 7) a teda je len jeho špeciálnym prípadom.

Implementácia: Použije sa rovnaká implementácia ako vo vzore 7) s tým, že všetky úlohy, ktoré majú aplikovaný tento vzor, patria iba do jednej skupiny.

Vzor 8) (Capability-Based Distribution)

Popis vzoru: Schopnosť distribúcie ľudských úloh zdrojom na základe špecifických schopností, ktoré zdroje majú. Schopnosti sú uložené pre každý zdroj zvlášť ako časť organizačného modelu. *Príklady* schopností: dĺžka praxe v odbore, získané licencie, výkonnosť.

Dôvod výberu: Tento vzor sa nám nezdá ako nevyhnutný, hoci je vhodný.

BPEL server: V správe popisujúcej úlohu sa pošlú informácie o vyžadovaných schopnostiach zdroja vo forme zoznamu usporiadaných dvojíc `[názov_schopnosti, hodnota_schopnosti]`. BPEL server si zároveň z databázy zisťuje informácie o schopnostiach zdrojov.

TM: Vhodná množina zdrojov, ktorým úloha môže byť pridelená, sa vyberie na základe SQL dotazu. Ten skontroluje, či dané zdroje majú všetky požadované schopnosti.

Dátové atribúty: O každom zdroji sa v databáze uchováva zoznam usporiadaných dvojíc `[názov_schopnosti, hodnota_schopnosti]`.

Vzor 13) (Distribution by Offer – Multiple Resources)

Popis vzoru: Tento vzor umožňuje distribúciu úlohy viacerým zdrojom v stave *ponúknutá*. Zdroje sú informované o úlohe, ale nie sú zaviazané úlohu vykonať. Preto úlohu môžu aj ignorovať, alebo ju ponúknuť inému zdroju.

Dôvod výberu: Tento vzor popisuje jednu z bežných, avšak nie nutných stratégií prideľovania úloh zdrojom. Dá sa čiastočne simulovať dvojicou vzorov 14) *Distribution by Allocation - Single Resource* a 27) *Delegation* (až na to, že úloha bude zdroju pridelená a nie ponúknutá).

TM: Každú úlohu, ktorá spĺňa tento vzor, ponúkne TM všetkým vhodným zdrojom. Ak niektorý zo zdrojov úlohu začne vykonávať, tak TM túto úlohu odstráni zo zoznamu úloh ostatných zdrojov.

TLB: TLB umožňuje zdrojom, aby menili vlastníka úlohy. Od vzoru 27) *Delegation* sa tento

vzor líši v tom, že nový vlastník získa úlohu v stave *ponúknutá* (a nie pridelená ako pri vzore 27) *Delegation*).

Vzor 12) (Distribution by Offer - Single Resource)

Popis vzoru: Tento vzor umožňuje distribúciu úlohy jednému zdroju v stave *ponúknutá*. Zdroj je informovaný o úlohe, ale nie je zaviazaný úlohu vykonať. Preto úlohu môže ignorovať, alebo ju ponúknuť inému zdroju.

Dôvod výberu: Tento vzor je zoslabením podmienky vzoru 13) a je podporovaný len tromi z porovnávaných nástrojov.

Všeobecné implementačné poznámky: Implementácia sa od vzoru 13) líši len v tom, že skupina má vždy iba jeden prvok.

Vzor 14) (Distribution by Allocation – Single Resource)

Popis vzoru: Možnosť priradiť úlohu konkrétnemu zdroju v stave *pridelená*[10]. Priradenie úlohy konkrétnemu zdroju sa deje až počas vykonávania biznis procesu, nie v čase jeho návrhu.

Dôvod výberu: Tento vzor popisuje jeden zo štandardných spôsobov prideľovania úloh zdrojom, keď zdroju sú úlohy iba pridelené a žiadne mu nie sú ponúknuté. Okrem toho tento vzor podporujú takmer všetky nástroje porovnávané v [8].

TM: Pošle úlohu vybranému zdroju v stave *pridelená*.

TLB: Zobrazí zdroju úlohu v stave pridelená.

Dátové atribúty: ID zdroja, ktorému má byť úloha priradená.

Vzor 15) (Random Allocation)

Popis vzoru: Schopnosť prideliť ľudské úlohy náhodne vybranému zdroju.

Vzor 16) (Round Robin Allocation)

Popis vzoru: Schopnosť prideliť ľudské úlohy zdroju vybranému na základe cyklického striedania sa zdrojov.

Vzor 17) (Shortest Queue)

Popis vzoru: Schopnosť prideliť ľudské úlohy zdroju vybranému na základe toho, že má najkratší zoznam svojich úloh.

Dôvod výberu vzorov 15 – 17: Tieto vzory síce nie sú veľmi často používané, ale sú ľahko implementovateľné. Je možné ich využiť ako jednoduchý aproximatívny prístup k *load balancing*-u úloh na zdroje. Navyše ich podporujú len dva z porovnávaných nástrojov zo stránky [8].

Implementácia vzorov 15 – 17: Po zistení, ktorým zdrojom môže byť posielaná úloha pridelená (na základe podmienok vyplývajúcich z iných aplikovaných vzorov), sa zo získaného zoznamu zdrojov vyberie pomocou TM ten, ktorý spĺňa funkciu popísanú v konkrétnom zo vzorov a tomu sa pridelí nasledovná úloha.

10 Workflow resource patterns umožňujú priradiť úlohu (z pôvodného stavu vytvorená) podľa konkrétneho aplikovaného vzoru v stavoch: ponúknutá jednému človeku, ponúknutá viacerým ľuďom a v stave pridelená. Toto sa dá vidieť aj na stavovom diagrame nášho systému (pozri obr. 4.3, str. 38).

Vzor 19) (Distribution on Enablement)

Popis vzoru: Možnosť oznámiť všetkým zdrojom, že inštancie úlohy sú k dispozícii, hneď ako je aj celá úloha pripravená na vykonávanie. Toto sa deje počas vykonávania biznis procesu, nie v čase návrhu. Tento vzor je aplikovaný na úlohu.

Príklad: Kamionista *K* môže odviesť náklad jabĺk do supermarketu *S* v meste *M* hneď ako do špedičného skladu prišli jablká od poľnohospodárov.

Dôvod výberu: Tento vzor popisuje jeden zo štandardných spôsobov prideľovania úloh zdrojom, keďže pripravenosť celej úlohy na vykonávanie je bežným spúšťačom, aby biznis proces vykonávacie prostredie pridelilo jej konkrétne inštancie jednotlivým zdrojom. Okrem toho tento vzor podporujú všetky nástroje porovnávané v [8].

TM: Pošle inštanciu úlohy vybranému zdroju v stave *pridelená*, hneď ako mu príde v správe od BPEL servera a je možné ju nejakému vhodnému zdroju prideliť.

TLB: Zobrazí zdroju úlohu v stave *pridelená*.

Vzory 21-23

Popis vzorov: Tieto vzory umožňujú väčšiu flexibilitu prideľovania úloh. Dávajú možnosť ovplyvniť výber vykonávaných úloh **zdrojom**, čo môže byť často žiadané.

Vzor 21) (Resource-Initiated Allocation):

Popis vzoru: Schopnosť urobiť záväzok postarať sa o úlohu bez toho, aby na nej zdroj musel začať ihneď po pridelení pracovať. Je možné, aby sa k vykonávaniu zaviazalo viacero zdrojov, z ktorých práve jeden naozaj úlohu vykoná a ostatným bude odobratá z ich zoznamov úloh.

Dôvod výberu: Je celkom bežné, že ľudia prijmú nejakú úlohu bez toho, aby na nej hneď začali pracovať (napr. preto, že zatiaľ pracujú na inej úlohe). Napriek predošlému argumentu, iba jeden nástroj z tých, ktoré sú porovnávané na stránke [8], podporuje tento vzor. Aj toto je priestor pre moju prácu.

TLB: Umožňuje zdroju zmeniť stav úlohy z *ponúknutá* na *pridelená*. Ak nejaký zo zdrojov, ktorý sa podujal úlohu vykonať, ju aj skutočne začne vykonávať, TLB o tom pošle informáciu do TM a ten odstráni úlohu zo zoznamov ich úloh.

Vzor 22) (Resource-Initiated Execution – Allocated Work Item):

Popis vzoru: Schopnosť zdroja vykonať úlohu, ktorá mu už bola predtým pridelená.

Dôvod výberu: Toto je vzor, ktorý logicky nadväzuje na vzor 21).

TLB: Umožňuje zdroju zmeniť stav úlohy z *pridelená* na *vykonávaná*.

Vzor 23) (Resource-Initiated Execution - Offered Work Item):

Popis vzoru: Schopnosť zdroja rozhodnúť sa pre niektorú z ponúkaných úloh a začať na nej ihneď pracovať.

Dôvod výberu: Tento vzor v podstate spája funkčnosť vzorov 21) a 22) do jedného kroku a je aj podporovaný väčšinou nástrojov porovnávaných na stránke [8].

TLB: Umožňuje zdroju zmeniť stav úlohy z *ponúknutá* na *vykonávaná*.

Vzor 24) (System-Determined Work Queue Content)

Popis vzoru: Schopnosť biznis proces vykonávacieho prostredia určiť obsah a poradie úloh, v ktorom budú zdroju prezentované na vykonanie. Poradie možno určiť napr. ako FIFO, LIFO, alebo podľa najbližšej lehoty na vykonanie úlohy.

Dôvod výberu: Tento vzor nie je veľmi potrebný, ale je vcelku ľahko implementovateľný a tiež ho podporujú len tri z nástrojov porovnané na stránke [8].

Všeobecné implementačné poznámky: Kritérium, podľa ktorého sa úlohy zoraďujú, sa nastavuje pre Task Manager (cez na to určený konfiguračný súbor). Všetky úlohy musia mať daný atribút, resp. tie, ktoré ho nemajú, sú zoradené za tými, ktoré ho majú.

TM: Filtruje a usporadúva zoznam úloh podľa podmienok kladených na atribúty.

Vzor 25) (Resource-Determined Work Queue Content)

Popis vzoru: Schopnosť zdroja určiť formát a obsah zoznamu svojich úloh na vykonanie.

Dôvod výberu: Tento vzor je jednoducho implementovateľný a pre používateľa predstavuje spríjemnenie práce.

Všeobecné implementačné poznámky: Všetky úlohy musia mať atribút, podľa ktorého sa úlohy zoraďujú alebo filtrujú[11] resp. tie, ktoré ho nemajú, sú zoradené za tými, ktoré ho majú (alebo sú odfiltrované – podľa logického kontextu). Niektoré atribúty sa posielajú priamo z BPEL servera (napr. ID úlohy, jej lehota, rola a pod.) iné sú nastavené v TM (napr. čas príchodu úlohy).

TLB: TLB zoradí úlohy po vybraní zoraďovacieho kritéria na základe jednoduchého triedenia poľa hodnôt vybraného atribútu.

Vzor 27) (Delegation)

Popis vzoru: Schopnosť zdroja prideliť úlohu, ktorá ešte nie je vykonávaná, inému zdroju. Každý zdroj, na ktorý je tento vzor aplikovaný, má právo delegovať svoje úlohy iným zdrojom.

Dôvod výberu: Táto funkčnosť je dosť často používaná.

TLB Umožňuje, aby zdroj, ktorému bola úloha *pridelená*, zmenil vlastníka úlohy. Úloha ostáva v stave *pridelená*.

Vzor 28) (Escalation)

Popis vzoru: Tento vzor umožňuje *biznis proces systému* aby úlohy, ktoré stoja v svojom vykonávaní na mieste, presunul inému vhodnejšiemu zdroju. Tento prípad všeobecne nastáva, ak sa prekročí určitá časová lehota zviazaná s úlohou. Môže mať však aj iné dôvody – napr. preventívny *load balancing* alebo dodržanie nejakej podmienky medzi úlohami.

Dôvod výberu: Tento vzor je potrebné podporovať, pretože zo vzorov je jediný, ktorý umožňuje biznis proces systému nové preusporiadanie úloh medzi viacerými zdrojmi.

Všeobecné implementačné poznámky: Pre úlohy je potrebné nejakým spôsobom zapísať lehoty – napr. informáciou v popise úlohy alebo na úrovni TM na základe určitých vopred stanovených kritérií. Lehoty môžu byť zapísané ako literály (t. j. presný dátum, alebo ako nejaká dĺžka – napr. 10 dní od začiatku vykonávania a pod.).

11 Zoraďovať sa dá iba podľa porovnateľných atribútov.

BPEL server: V správe o úlohe môže byť ako jeden z atribútov poslaný platná lehota (teda taká, ktorá je po čase príchodu do systému). Iné lehoty budú ignorované.

TM: Lehoty jednotlivých eskalácií si udržiava v usporiadanom poli dvojíc podľa času: `[čas_eskalácie, identifikátor_úlohy]`. Na zistenie, že už uplynula nejaká lehota, TM používa časovač, ktorý má nastavený vždy najnižší čas z poľa lehôt. Počas samotného eskalovania TM odstráni úlohu zo zoznamu úloh daného zdroja v stave *vykonávaná* a *ponúkne* resp. *pridelí* ju inému zdroju. Samotná eskalácia úlohy inému zdroju, alebo skupine sa dá implementovať viacerými spôsobmi:

1. Mať užívateľa s právami *správca úloh*, ktorý by mohol úlohy, ktoré prekročili stanovený časový limit, presunúť ľubovoľnému zdroju, ktorý by spĺňal ostatné vzory aplikované na úlohu.
2. Mať definovanú stromovú hierarchiu v XML súbore, ktorej vrcholy by tvorili ľudia a / alebo roly. Ak by nastala eskalácia, tak by sa TM pokúsil priradiť, resp. ponúknuť úlohu vrcholu, ktorý je o úroveň vyššie ako predošlý vrchol. Po prípadnom neúspechu eskalácie v koreni stromu by celá eskalácia skončila ako neúspešná a úloha by prešla do stavu *nedokončiteľná*.

Dátové atribúty: Usporiadané pole dvojíc `[čas_eskalácie, identifikátor_úlohy]` podľa času.

Vzor 29) (Deallocation)

Popis vzoru: Schopnosť zdroja (alebo skupiny zdrojov) vzdať sa pridelenej, ale ešte nevykonávanej úlohy a poskytnúť ju iným zdrojom.

Dôvod výberu: Tento vzor je praktický a ľahko implementovateľný.

TLB: Umožňuje zdroju, na ktorý je aplikovaný tento vzor, aby zmenil stav nejakej úlohy zo svojho zoznamu úloh z *pridelená* na *dealokovaná*.

TM: Na správu, že nejaká úloha je *dealokovaná* reaguje TM tak, že vyhodnotí, akým iným zdrojom úlohu môže ponúknuť a zmení jej stav v zozname úloh pre vyhovujúce zdroje na *ponúknutá*. Toto isté TM robí aj pri každej úlohe, ktorá práve prišla do systému.

Vzor 30) (Stateful Reallocation)

Popis vzoru: Schopnosť zdroja prideliť úlohu, ktorú práve vykonáva, inému zdroju bez straty informácií, ktoré už boli čiastočnou prácou pôvodného zdroja vyprodukované.

Dôvod výberu: Tento vzor nie je nevyhnutný, ale spríjemňuje užívateľovi pracujúcemu s konkrétnymi procesmi jeho prácu.

Všeobecné implementačné poznámky: Informácie posielané spolu s úlohou sa týkajú jej atribútov (aplikované vzory, lehoty a pod.) – tieto sú automaticky súčasťou úlohy a nie je preto problém s ich posielaním. Ostatné informácie v elektronickej forme sú vlastne len súbormi. Tieto možno buď poslať definovaním url, na ktorom sa nachádzajú resp. ich fyzicky skopírovať z miesta, kde ich mal uložené pôvodný zdroj na miesto, kde ich chce mať uložené nový zdroj.

TLB: Umožní zdroju poslať aj súbory inému zdroju. Nový zdroj bude mať pri pridelenej úlohe aj prílohu so súbormi.

TM: Ak sú súčasťou informácií o úlohe aj súbory, tieto si TM skopíruje na nevyhnutný čas do svojho dočasného priečinka a hneď, ako sa úlohy ujme nový zdroj, tak mu ich presunie.

Dátové atribúty: Presúvajú sa iba súbory, ktoré boli potrebné na vykonanie úlohy, ale nie sú jej súčasťou (napr. medzivýpočty).

Vzor 31) (Stateless Reallocation)

Popis vzoru: Je to vzor takmer identický so vzorom 27) *Delegation* s tým rozdielom, že úloha už bola v procese vykonávania. Žiadne z informácií, ktoré súvisia s vykonávaným stavom úlohy, sa neprenášajú s úlohou ďalej k jej novému vlastníkovi. To znamená, že úloha je u nového vlastníka reštartovaná.

Dôvod výberu: Obdobne ako pri vzore 27) *Delegation*.

TLB: Umožňuje *zdroju*, na ktorý je aplikovaný tento vzor, aby zmenil stav úlohy z *vykonávaná* na *pridelená* a presunul úlohu inému zdroju. Pritom treba zabezpečiť „zahodenie“ informácií.

Vzor 32) (Suspension/Resumption)

Popis vzoru: Možnosť zdroja signalizovať biznis proces vykonávaciemu prostrediu dočasné zastavenie vykonávania ľubovoľnej úlohy a obrátiť svoju pozornosť na iné úlohy. Úloha ostáva v zozname úloh zdroja. Zdroj môže úlohu niekedy v budúcnosti reštartovať.

Dôvod výberu: Tento vzor nie je nevyhnutný, ale spríjemňuje užívateľovi prácu a je ľahké ho implementovať. Navyše ho podporujú len tri z nástrojov porovnané na stránke [8].

Všeobecné implementačné poznámky: O stave úlohy si stačí uchovávať len ďalšiu informáciu.

TLB: Umožní zdroju zmeniť stav úlohy z *vykonávaná* na *odložená.*

TM: Eviduje zmenu stavu úlohy a umožní zdroju, aby mohol pracovať na inej úlohe (previedol ju do stavu vykonávaná).

Vzor 33) (Skip)

Popis vzoru: Schopnosť zdroja preskočiť pridelenú úlohu a označiť ju ako vykonanú.

Dôvod výberu: Tento vzor umožňuje ignorovať niektoré menej podstatné úlohy a je ľahko implementovateľný, pretože mení iba stav úlohy. Navyše, bez ohľadu na tento vzor, náš systém musí umožňovať, aby zdroje mohli informovať, že úloha je vykonaná.

TLB: Umožňuje zdroju s týmto vzorom, aby zmenil aktuálny stav úlohy na *vykonaná*.

Vzor 42 (Simultaneous Execution)

Popis vzoru: Schopnosť zdroja vykonávať súčasne viacero úloh.

Dôvod výberu: Je to vhodný a ľahko implementovateľný vzor.

TLB: Musí len umožniť zobrazovanie viacerých úloh zdroju (a nie len jednu), ktoré môžu byť aktuálne vykonávané.

4.5.2 Nepodporované vzory

V tejto časti sú popísané vzory, ktoré nebudú implementované spolu s odôvodnením.

Vzor 3) (Deferred Allocation)

Popis vzoru: Schopnosť oddialiť určenie zdroja, ktorý vykoná danú úlohu až na čas, keď sa bude biznis proces vykonávať. Tento spôsob nepriameho priraďovania je možné docieliť napr. pomocou vopred dohodnutej premennej, na základe hodnoty ktorej možno za behu určiť identitu zdroja, ktorý vykoná úlohu. Premenná určujúca identitu zdroja sa dá podľa potreby meniť počas vykonávania biznis procesu.

Dôvod nevybratia vzoru: Z podrobného popisu vzoru vyplýva, že podmienky na základe ktorých by sa vyhodnocoval počas vykonávania inštancie biznis procesu konkrétny zdroj, môžu byť značne rozmanité a preto komplikované na implementáciu.

Vzor 4) (Authorization)

Popis vzoru: Schopnosť určiť rozsah privilégií, ktoré bude zdroj mať, čo sa vykonávania procesu týka. V podstate ide o rozsah privilégií, aké akcie bude zdroj môcť s úlohami vykonať.

Tento vzor má formu množiny relácií medzi zdrojmi a privilégiami, ktoré majú privilégiá vo vzťahu ku konkrétnemu procesu. Vymenujeme niektoré z privilégií:

- **choose** – schopnosť vybrať si nasledujúcu úlohu, ktorú zdroj vykoná,
- **view offers** – schopnosť vidieť všetky ponúknuté úlohy v celom biznis procese,
- **chained execution –** schopnosť pracovať s úlohami ako pri vzore chained execution,
- **delegate** – schopnosť delegovať úlohy.

Dôvod nevybratia vzoru: Z popisu vzoru (najmä z popisu privilégií) je zrejmé, že tento vzor je akoby filter, ktorý v prípade, ak je na nejaký zdroj aplikovaných viacero vzorov, definuje, ktoré z týchto vzorov budú povolené. Toto je síce vhodné, avšak nepridáva to žiadnu novú funkcionalitu. Rovnaký efekt sa dá dosiahnuť aj tým, že dizajnér vhodne vyberie množinu vzorov, ktoré budú na zdroj aplikované. Pre práve spomenutú skutočnosť nebudeme tento vzor podporovať.

Vzor 9) (History-Based Distribution)

Popis vzoru: Schopnosť distribúcie úloh na základe ich predošlej histórie vykonávania.

Dôvod nevybratia vzoru: Hlavný dôvod je, že vlastnosti, ktoré sa týkajú histórie vykonávania a majú sa na ich základe prideľovať zdrojom úlohy, môžu byť prakticky ľubovoľné. Pri každom vyhodnocovaní funkcie by sa muselo pristupovať do databázy a daná funkcia by sa musela posielať v správe (čo by mohli byť napr. štandardné SQL dotazy). Úplne všeobecná implementácia by preto nebola možná, lebo databáza má len obmedzené množstvo atribútov. Bola by však možná implementácia nad vopred stanovenou množinou atribútov.

Vzor 10) (Organisational Distribution)

Popis vzoru: Schopnosť distribúcie úloh na základe ich pozície vrámci organizácie a jej vzťahu k ostatným zdrojom.

Dôvod nevybratia vzoru: Tento vzor je podporovaný malým počtom nástrojov a aj tými veľmi rôzne. Organizačná štruktúra by sa dala modelovať pomocou stromu alebo grafu, ktorý by sa uchovával vo forme nejakého súboru ako súčasť konkrétneho procesu. Bolo by potrebné definovať relácie medzi jednotlivými vrcholmi (ak by sa vyžadovali aj iné relácie ako otec, synovia, bratia,...). Toto by vo všeobecnosti bolo značne implementačne náročné v porovnaní s tým, čo by to prinášalo.

Vzor 11) (Automatic Execution)

Popis vzoru: Tento vzor popisuje schopnosť úlohy, aby bola vykonaná bez potreby ľudského zdroja.

Dôvod nevybratia vzoru: Tento vzor sa zo svojej podstaty netýka podpory ľudí v biznis procesoch. Okrem toho túto funkcionalitu podporuje už aj samotný jazyk BPEL pomocou aktivity `<invoke>`.

Vzor 18) (Early Distribution)

Popis vzoru: Schopnosť *biznis proces systému* ponúknuť, resp. distribuovať úlohy už pred tým, ako sú dostupné na vykonávanie.

Dôvod nevybratia vzoru: Tento vzor nie je príliš často využiteľný a neprináša žiadnu podstatnú funkčnosť.

Vzor 20) (Late Distribution)

Popis vzoru: Schopnosť *biznis proces systému* ponúknuť, resp. distribuovať úlohy až potom, ako sú dostupné v systéme na vykonávanie.

Dôvod nevybratia vzoru: Tento vzor neprináša žiadnu podstatnú funkčnosť.

Vzor 34) (Redo)

Popis vzoru: Schopnosť zdroja nanovo vykonať úlohu *x* a všetky úlohy, ktoré boli po vykonaní úlohy *x* vykonané a závisia od nej. Ide vlastne o akúsi možnosť navrátenia sa do určitého bodu vykonávania biznis procesu a jeho opakovaného vykonania. Toto môže byť motivované napr. nedostatočným spracovaním úloh alebo tým, že boli získané lepšie údaje a pod.

Dôvod nevybratia vzoru: Hoci tento vzor prináša zaujímavú funkčnosť, jeho implementácia by bola spojená s množstvom technických problémov.

Vzor 35) (Pre-Do)

Popis vzoru: Schopnosť zdroja vykonať úlohu *x* pred úlohou *y*, hoci úloha *x* logicky, časovo alebo z iného dôvodu nasleduje až po úlohe *y*.

Dôvod nevybratia vzoru: Tento vzor neponúka žiadnu výraznú pridanú hodnotu. Okrem toho je s ním spojených viacero komplikácií tak, ako sú podrobne popísané na stránke [8].

Vzor 36) (Commencement on Creation)

Popis vzoru: Schopnosť zdroja vykonať úlohu hneď ako je vytvorená. To znamená, že kroky vytvorenia, pridelenia a začatia vykonávania úlohy sa zlejú do jedného.

Dôvod nevybratia vzoru: Tento vzor vyžaduje, aby kroky *ponúknutie, alokácia a spustenie úlohy* tvorili spolu jednu atomickú akciu. Toto je pre ľudí nemožné dodržať, pretože neexistuje spôsob, ako ich prinútiť, aby za každých okolností začali pracovať hneď na pridelenej úlohe.

Vzor 37) (Commencement on Allocation)

Popis vzoru: Schopnosť zdroja vykonať úlohu hneď ako mu je pridelená. To znamená, že kroky pridelenia a začatia vykonávania úlohy sa zlejú do jedného.

Dôvod nevybratia vzoru: V tomto vzore je rovnaký problém s ľuďmi ako vo vzore 36).

Vzor 38) (Piled Execution)

Popis vzoru: Schopnosť zdroja vykonávať v jednom slede všetky inštancie jedného typu úlohy v danom biznis procese (prípadne aj vo viacerých).

Dôvod nevybratia vzoru: Tento vzor nie je podporovaný žiadnym z porovnávaných nástrojov a problém by bol aj s manažovaním súčasného prideľovania inštancií úloh z rôznych biznis procesov.

Vzor 39) (Chained Execution)

Popis vzoru: Schopnosť zdroja vykonávať v jednom slede všetky úlohy, ktoré má v svojom zozname. Teda ukončenie jednej úlohy je spúšťačom zmeny stavu nasledujúcej úlohy na vykonávaná. Idea tohto vzoru je využiť zdroj tak, aby mal čo najmenší možný počet prestojov, kedy nemá prácu.

Dôvod nevybratia vzoru: Zmena stavu úlohy človekom na stav *vykonávaný* je zanedbateľne krátka činnosť, a preto tento vzor stráca reálne opodstatnenie.

Vzor 40) (Configurable Unallocated Work Item Visibility)

Popis vzoru: Schopnosť biznis proces systému konfigurovať viditeľnosť zatiaľ nepridelených úloh v celom systéme pre zdroje.

Dôvod nevybratia vzoru: Najzaujímavejšou informáciou pre zdroje, ktoré sa zaujímajú aj o iné ako svoje úlohy, je informácia o ponúkaných úlohách. Túto funkcionalitu však už zabezpečujú vzory 12) *Distribution by Offer - Single Resource* a 13) *Distribution by Offer – Multiple Resources*. Informácie o iných stavoch úloh nie sú pre zdroje veľmi zaujímavé.

Vzor 41) (Configurable Allocated Work Item Visibility)

Popis vzoru: Schopnosť vykonávacieho prostredia pre biznis proces konfigurovať viditeľnosť už pridelených úloh v celom systéme pre zdroje.

Dôvod nevybratia vzoru: Dôvod je rovnaký ako pri vzore 40) *Configurable Unallocated Work Item Visibility*.

Vzor 43) (Additional Resources)

Popis vzoru: Schopnosť zdroja požiadať biznis proces systém o ďalšie zdroje, ktoré budú spolu s ním pracovať na jednej úlohe.

Dôvod nevybratia vzoru: Pri implementácii by vznikali viaceré problémy – napr. s tým, že kedy je možné zmeniť stav úlohy. (Mohol by sa napr. určiť šéf zdrojov, ktoré pracujú spolu na úlohe a pod.)

4.6 Prínosy nášho systému

V tejto časti uvádzame, ktoré časti nášho systému boli inšpirované architektúrou yawl systému (popísaný v časti 1.4.4, str. 16), porovnávame náš systém s prácami popísanými v častiach 3.1, str. 24 a 3.2, str. 29 a popisujeme jeho prínosy.

Z yawl systému sme sa inšpirovali dvoma vecami – komponentovou architektúrou a konkrétnym komponentom YAWL worklist handler, ktorý rovnako ako náš komponent TM priraďuje prácu jednotlivým zdrojom.

Podobne ako v práci [14], ani náš prístup nevyžaduje, aby pre podporu práce s ľudskými úlohami tieto schopnosti muselo nejakým spôsobom podporovať vykonávacie prostredie BPEL (napr. interpretovaním syntaktického rozšírenia jazyka). Ďalej myšlienka generovania HTML formulárov v našom systéme z XML súborov s definovanou schémou je podobná tej, aká bola využitá v práci [15].

Prínosy nami navrhnutého systému:

- Náš systém si dal za cieľ priamo podporovať vybranú podmnožinu akademickej iniciatívy workflow resource patterns, čo nie je ambíciou žiadneho z popisovaných systémov (hoci niektoré zo systémov ich nepriamo podporujú).
- Pred používateľom skrýva implementačné detaily súvisiace s podporou ľudských úloh (správu ich stavov, dodržiavanie aplikovaných vzorov a iné).
- Vďaka použitiu všeobecne akceptovanej technológie webovej služby možno náš systém používať na ľubovoľnej IP adrese – t. j. lokálne, vo firemnom intranete ako aj v medzifiremnej sieti. Preto môže byť nasadenie nášho systému oveľa jednoduchšie ako pri desktopových aplikáciách, ktoré zväčša musia byť inštalované na každom počítači.
- Pre dizajnérov biznis procesov ponúka šablóny bežných úloh, ale ponecháva im aj možnosť definovať si vlastné úlohy.

4.6.1 Porovnanie nášho systému s BPEL4People

K **výhodám rozšírenia BPEL4People oproti nášmu systému** patrí:

- Je navrhnuté skupinou významných softvérových firiem, vďaka čomu je pravdepodobné, že bude v blízkej budúcnosti všeobecne podporované vykonávacími prostrediami pre BPEL.

K **nevýhodám rozšírenia BPEL4People oproti nášmu systému** patrí:

- Špecifikácie tvoriace BPEL4People boli navrhované len s ohľadom na praktické potreby, ale nie na akademické práce (teda ani vzhľadom na workflow resource patterns).
- Pre úspešné vykonávanie kódu zapísaného pomocou rozšírenia BPEL4People nestačí väčšina terajších vykonávacích prostredí pre BPEL a je potrebné, aby boli buď rozšírené, (napr. pomocou zásuvného modulu), alebo aby využívali externé systémy na ich interpretáciu, resp. preklad (podobne ako systém VieBOP popísaný v časti 3.2.1, str. 29).

K **výhodám nášho systému oproti rozšíreniu BPEL4People** patrí:

- Je navrhnutý s cieľom podporovať konzistentnú podmnožinu workflow resource patterns.

K **nevýhodám nášho systému oproti rozšíreniu BPEL4People** patrí:

- Výber vzorov je špecifický pre potreby UK, a tak nebude nikdy štandardom.

Nasledovná tabuľka porovnáva, ktoré z workflow resource patterns podporuje BPEL4People, a ktoré náš systém.

skupina vzorov	vzor	podpora BPEL4People	podpora naším systémom
creation patterns	1) Direct Allocation	X	X
	2) Role-Based Allocation	X	X
	3) Deferred Allocation	X	-
	4) Authorisation	-	-
	5) Separation of Duties	X	X
	6) Case Handling	-	X
	7) Retain Familiar	-	X
	8) Capability-Based Allocation	-	X
	9) History-Based Allocation	-	-
	10) Organisational Allocation	-	-
	11) Automatic Execution	X	-
Push Patterns	12) Distribution by Offer - Single Resource	X	X
	13) Distribution by Offer - Multiple Resources	X	X
	14) Distribution by Allocation - Single Resource	X	X
	15) Random Allocation	-	X
	16) Round Robin Allocation	-	X
	17) Shortest Queue	-	X
	18) Early Distribution	*	-
	19) Distribution on Enablement	X	X
	20) Late Distribution	-	-
Pull patterns	21) Resource-Initiated Allocation	X	X
	22) Resource-Initiated Execution - Allocated Work Item	X	X

skupina vzorov	vzor	podpora BPEL4People	podpora naším systémom
	23) Resource-Initiated Execution - Offered Work Item	X	X
	24) System-Determined Work Queue Content	-	X
	25) Resource-Determined Work Queue Content	-	X
	26) Selection Autonomy	X	-
Detour Patterns	27) Delegation	X	X
	28) Escalation	X	X
	29) Deallocation	X	X
	30) Stateful Reallocation	X	X
	31) Stateless Reallocation	X	X
	32) Suspension/ Resumption	X	X
	33) Skip	X	X
	34) Redo	-	-
	35) Pre-Do	-	-
Auto-start Patterns	36) Commencement on Creation	-	-
	37) Commencement on Allocation	-	-
	38) Piled Execution	-	-
	39) Chained Execution	-	-
Visibility Patterns	40) Configurable Unallocated Work Item Visibility	-	-
	41) Configurable Allocated Work Item Visibility	-	-
Multiple Resource Patterns	42) Simultaneous Execution	X	X
	43) Additional Resources	-	-

Tabuľka 4.2: Porovnanie podpory workflow resource patterns v BPEL4People a našom systéme.

Vysvetlivky:

v stĺpci „podpora BPEL4People“ znamenajú značky: X – úplná podpora daného vzoru; * čiastočná podpora daného vzoru; - nepodporovaný vzor;

v stĺpci „podpora našim systémom“ znamenajú značky: X – vzor bude určite podporovaný; - vzor nebude podporovaný

4.7 Ako nami navrhnuté riešenie splnilo požiadavky

Z požiadaviek popísaných v časti 4.1, str. 32 sme v našom návrhu a kostre implementácie systému splnili jednotlivé body nasledovne.

Náš systém umožňuje rozšíriť biznis proces systémy využívajúce jazyk BPEL o podporu ľudských úloh, ako sme si to vytýčili v bode 1, tým že navrhuje implementáciu vybraných vzorov z *workflow resource patterns*, ktorých podmnožinu sme vybrali v bode 2, resp. v niektorých prípadoch ju systém aj realizuje. Vďaka tomu, že pre komunikáciu s vykonávacím prostredím BPEL využívame webovú službu, tak sa nám podarilo splniť aj požiadavku č. 3, aby náš systém bol nezávislý na vykonávacom prostredí jazyka BPEL. V realizácii požiadavky popísanej v poslednom, 4. bode našej špecifikácie hovoriacom o čo najjednoduchšej použiteľnosti a nasaditeľnosti nášho systému v kombinácii s vykonávacím prostredím jazyka BPEL vidíme isté rezervy. Najmä z pohľadu používateľa – programátora sa od neho v aktuálnom stave systému vyžaduje pomerne hlboká znalosť technického zápisu ľudských úloh v BPEL procese na to, aby náš systém bol pre neho použiteľný.

Pri zamýšľaní sa nad ďalšou možnou implementáciou systému sme narazili na rôzne zaujímavé problémy. Jedným z nich je problém, ako je možné vo všeobecnosti zistiť, či už celá inštancia biznis procesu skončila. Toto je napr. potrebné vedieť pre „zahadzovanie“ informácii, ktoré sa týkajú len jednej inštancie procesu. Ďalším problémom môže byť vyriešenie paralelného vykonávania viacerých biznis procesov (resp. ich inštancií) naraz – najmä spravovanie informácií súvisiacich s úlohami a zdrojmi. Zdroje totižto môžu byť zapojené do viacerých biznis procesov naraz, no úlohy z jednotlivých procesov by sa nemali medzi nimi pomiešať – t. j. mali by byť počas ich celého životného cyklu obhospodarované len v rámci jednej inštancie biznis procesu.

5 Záver

Predkladaná práca mala dva hlavné ciele:

1. Urobiť prehľad a porovnanie v súčasnosti používaných jazykov na popis biznis procesov a posúdiť, ktorý z nich by mohol poskytnúť najvhodnejší základ na implementáciu systému na podporu biznis procesov na Univerzite Komenského v Bratislave.
2. Navrhnúť a čiastočne implementovať čo najvšeobecnejší systém podporujúci využívanie biznis procesov s podporou ľudí na UK.

Prvý z cieľov sa nám podarilo uspokojivo naplniť podrobným popisom možností jednotlivých jazykov, ich vzájomným porovnaním a analýzou vzhľadom na *workflow resource patterns*.

Druhý z cieľov sme z časových dôvodov splnili len čiastočne – systém sme síce navrhli, ale jeho implementácia ostala v štádiu, v ktorom ešte nie je použiteľná a nie je možné ju žiaľ predviesť ani vo forme prototypu.

K prínosom našej práce patria najmä podrobná analýza podpory *workflow resource patterns* rozšírením BPEL4People a najmä návrh vlastného systému rozširujúceho jazyk BPEL o podporu ľudských úloh.

Možnosti rozpracovania práce

V teoretickej časti práce by bolo možné popísať aj menej významné jazyky pre biznis procesy (ako napr. BPML, WS-CDL) a porovnať ich s ostatnými.

V praktickej časti práce by bolo potrebné v prvom rade dokončiť implementáciu všetkých komponentov navrhnutého systému do takej miery, aby systém bolo možné otestovať aspoň vo forme prototypu.

Slovník pojmov

V tomto slovníku sú jednotlivé pojmy označené tučným písmom. Ak sa niektoré pojmy v svojej definícii odkazujú na iné pojmy zo slovníka, tak tieto odkazované pojmy sú zapísané kurzívou.

- **biznis proces** – Je to popis krokov vykonávania nejakých *úloh* a vzťahov medzi nimi dostatočne detailne nato, aby mohol byť priamo vykonaný *vykonávacím prostredím pre biznis proces.* Je reprezentovaný orientovaným grafom, kde *úlohy* sú vrcholmi a hrany reprezentujú prechody medzi nimi s definovanými podmienkami.
- **BPM** (Business Process Management) – Je to manažment komplexných interakcií medzi ľuďmi, aplikáciami, strojmi a rôznymi technológiami.
- **inštancia biznis procesu** – Je to ľubovoľné vykonanie konkrétneho *biznis procesu*. Od biznis procesu sa líši tým, že každá inštancia biznis procesu môže byť iná (napr. sa môžu rôzne vyhodnotiť podmienky v biznis procese). Príklad: Ak je v biznis procese nejaké vetvenie s vetvami A a B, tak v jednej inštancii procesu sa môže vykonať vetva A, v inej inštancii toho istého procesu sa vykoná vetva B.
- **inštancia úlohy** – Je ňou každé vykonanie nejakej *úlohy*.
- **JSP** (java server pages) – Technológia pre tvorbu interaktívnych webových stránok využívajúca na serverovej strane platformu java. Pred jej prvým použitím ju web server prekladá na *servlet*.
- **load balancing** – Ak máme v systéme viacero zdrojov (či procesov, ľudí alebo iných) a *úlohy* pre túto skupinu, tak snaha rozumne rozdeliť úlohy medzi zdroje definuje pojem load balancing.
- **ľudská úloha** – Úloha ako súčasť *biznis procesu*, ktorá je určená pre ľudí. Príkladom môže byť vykonanie telefonátu na nejaký úrad.
- **okľuka** (workaround) – Je to spôsob ako realizovať nejakú funkčnosť alebo vlastnosť v jazyku napriek tomu, že ním nie je priamo podporovaná.
- **otvorený zdrojový kód** (opensource) – Je to každý program alebo knižnica, ku ktorej je verejne prístupný aj jej zdrojový kód.
- **servlet** – Program napísaný v jazyku java, ktorý beží na webovskom alebo aplikačnom serveri a generuje na základe vstupných argumentov výstupný dokument – väčšinou webovú stránku.
- **úloha** – Je to popis nejakej jednotky práce. V našej práci delíme úlohy na *ľudské* a strojové. Príkladom môže byť vykonanie telefonátu na nejaký úrad (pre *ľudskú úlohu*), resp. vykonanie automatickej zálohy dokumentov (pre strojovú úlohu).
- **úloha spĺňa vzor** – Úloha spĺňa vzor *V*, keď je na ňu vzor *V* aplikovaný, t. j. keď má tento vzor *V* uvedený v svojom zozname vzorov a dodržiavajú sa vlastnosti vyplývajúce zo vzoru. Príklad: Nato, aby úloha *U* spĺňala vzor 2) Role-Based Distribution, tak musí byť pre úlohu *U* zachované, že môže byť pridelená len niektorému zo zdrojov, ktoré sú v zozname zdrojov tvoriacom danú rolu.
- **vykonávacie prostredie pre biznis proces** – Nástroj na vykonávanie postupností *úloh* zapísaných v *biznis procese*.
- **vykonávacie prostredie BPEL** – *Vykonávacie prostredie pre biznis procesy* zapísané

v jazyku BPEL.

- **webová služba** – Je to softvérový systém umožňujúci spoluprácu medzi dvoma počítačovými programami po sieti. Často sú takto umožnené len vzdialené volania procedúr.
- **whitepaper** – V oblasti informatiky sa pod týmto pojmom myslí väčšinou dokument, ktorý robí náhľad na pripravovaný štandard alebo koncept, alebo vysvetľuje stručne už nejaký existujúci koncept.
- **wsdl** (Web Services Description Language) – Jazyk na popis operácií *webových služieb* – najmä ich argumentov a chybových stavov zapísaných vo forme správ spĺňajúcich pridruženú *XML schému*. Používa sa predovšetkým pre popis rozhraní systémov, ktoré možno vzdialene volať cez sieť.
- **workflow** – V našej práci tento výraz používame ako synonymum k výrazu *biznis proces*.
- **XPath** (XML path language) – jazyk obsahujúci výrazy pre adresovanie konkrétnych častí a uzlov v XML dokumente a pre jednoduché operácie pracujúce nad časťami XML dokumentu.
- **XML schéma** – Je to jeden z viacerých XML jazykov pre popis schémy XML dokumentu. Umožňuje definovať typy elementov a rôzne vzťahy medzi nimi.
- **xsd** (XML Schema Definition) – Je to termín pre inštanciu XML schémy.
- **XSLT** (eXtensible Stylesheet Language Transformations) – Je to XML jazyk umožňujúci transformovať vstupný XML dokument na ľubovoľný iný dokument (väčšinou opäť na iný XML alebo HTML dokument).
- **zdroj** – Zdrojom sa v našej práci myslí ľubovoľná entita schopná vykonať prácu. Od kapitoly 3 sa ňou myslí človek.
- **zdroj spĺňa vzor** – Zdroj spĺňa vzor *V*, keď je naň vzor *V* aplikovaný, t. j. keď má tento vzor *V* uvedený v svojom zozname vzorov a dodržiavajú sa vlastnosti vyplývajúce zo vzoru. Príklad: Nato, aby zdroj *Z* spĺňal vzor 30) Stateful Reallocation, tak zdroj *Z* musí umožňovať, aby jemu pridelené úlohy bolo možné presunúť v rozpracovanom stave inému zdroju a skopírovať s nimi aj čiastočne rozpracované pomocné súbory a pod.

Zoznam bibliografických odkazov

[1] *Web Services Business Process Execution Language Version 2.0*, OASIS, [online]. Dátum aktualizácie/revízie: apríl 2007, [cit. august 2007].
Dostupné na internete: <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf>

[2] *WS-BPEL 2.0 Extensions for Sub-Processes* [online]. Dátum aktualizácie/revízie: október 2005, [cit. január 2007].
Dostupné na internete: <http://www-128.ibm.com/developerworks/library/specification/ws-bpelsubproc/>

[3] *Business Process Modeling Notation Specification Version 1.0*, Object Management Group, [online]. Dátum aktualizácie/revízie: február 2006, [cit. január 2007].
Dostupné na internete: <http://www.omg.org/docs/dtc/06-02-01.pdf>

[4] *Business Process Model and Notation (BPMN) 2.0 Request For Proposal,* Object Management Group, [online]. Dátum aktualizácie/revízie: marec 2007, [cit. august 2007]. Dostupné na internete: <http://www.bpmn.org/Documents/BPMN%202-0%20RFP%2007-06-05.pdf>

[5] *XML Process Definition Language Version 2.0*, Workflow Management Coalition [online]. Dátum aktualizácie/revízie: október 2005, [cit. január 2007].
Dostupné na internete: <http://www.wfmc.org/standards/docs/TC-1025_xpdl_2_2005-10-03.pdf>

[6] Nathaniel Palmer - executive director of WfMC, *Understanding the BPMN-XPDL-BPEL value chain* [online]. Dátum aktualizácie/revízie: december 2006, [cit. február 2007].
Dostupné na internete: <http://www.bijonline.com/index.cfm?section=article&aid=806#>

[7] W.M.P. van der Aalst, L. Aldred, M. Dumas, A.H.M. ter Hofstede, *Design and implementation of the YAWL system*; [online] Riga, Latvia; Springer Verlag; dátum aktualizácie/revízie: jún 2004, [cit. február 2007].
Dostupné na internete: <http://www.bpm.fit.qut.edu.au/about/docs/yawl_system.pdf>
Dostupné tiež na: <http://citeseer.ist.psu.edu/vanderaalst04design.html>

[8] Eindhoven University of Technology (led by Professor Wil van der Aalst) and Queensland University of Technology (led by Associate Professor Arthur ter Hofstede); *Workflow Patterns* [online]. Dátum aktualizácie/revízie: 2007, [cit. október 2007]. Dostupné na internete: <http://www.workflowpatterns.com>

[9] *YAWL – YAWL: Yet Another Workflow Language (Revised version)* [online]. Dátum aktualizácie/revízie: september 2003, [cit. február 2007].

Dostupné na internete: <yawlfoundation.org/documents/yawlrevtech.pdf>

[10] *YAWL* [online]. Dátum aktualizácie/revízie: február 2007, [cit. február 2007]. Dostupné na internete: <http://sourceforge.net/projects/yawl/>

[11] Martin Vasko and Schahram Dustdar, *A view based analysis of workflow modeling languages*; Vienna University of Technology, Distributed Systems Group, Information Systems Institute; IEEE Computer Society; [online]. Dátum aktualizácie/revízie: február. 2006, [cit. Február 2007]; ISBN 0-7695-2513-X, s. 293-300;
Dostupné tiež v pdf formáte na:
<http://www.infosys.tuwien.ac.at/Staff/sd/papers/A%20view%20based%20analysis%20of%20workflow%20modeling%20languages.pdf>

[12] Ashish Agrawal et al.; *Specification Web Services Human Task (WS-HumanTask), Version 1.0*; [online]. Dátum aktualizácie/revízie:jún 2007, [cit. október 2007]. Dostupné na internete:<https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/docs/library/uuid/cfab6fdd-0501-0010-bc82-f5c2414080ed>

[13] Ashish Agrawal et al.; *Specification WS-BPEL Extension for People (BPEL4People)*, *Version 1.0; [online]*. Dátum aktualizácie/revízie:jún 2007, [cit. október 2007]. Dostupné na internete: <https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/30c6f5b5-ef02-2a10-c8b5-cc1147f4d58c>

[14] Ta'id Holmes, Martin Vasko, Schahram Dustdar; *VieBOP: Extending BPEL Engines with BPEL4People,* Vienna, Austria, Distributed Systems Group, Institute of Information Systems, Vienna University of Technology; IEEE Computer Society; Dátum aktualizácie/revízie: február 2008, [cit. Marec 2008]; ISBN 978-0-7695-3089-5, s. 547-555
Dostupné tiež v pdf formáte na: <http://www.infosys.tuwien.ac.at/Staff/sd/papers/VieBOP.pdf>

[15] Y. S. Kuo, Lendle Tseng et al.; *An XML Interaction Service for Workflow Applications,* Taiwan, Institute of Information Science, Academia Sinica; ACM; Dátum aktualizácie/revízie: október 2006, [cit. Apríl 2008]; ISBN 1-59593-515-0, s. 53-55
Dostupné tiež v pdf formáte na:
<http://portal.acm.org/ft_gateway.cfm?id=1166177&type=pdf&coll=Portal&dl=GUIDE&CFID=22679850&CFTOKEN=12765708>

[16] Dipanjan Chakraborty and Hui Lei; *Pervasive Enablement of Business Processes;* University of Maryland Baltimore County, IBM T. J. Watson Research Center; IEEE Computer Society; Dátum aktualizácie/revízie: marec 2004, [cit. Apríl 2008]; ISBN 0-7695-2090-1, s. 87-98
Dostupné tiež v pdf formáte na:

<http://csdl.computer.org/dl/proceedings/percom/2004/2090/00/20900087.pdf>

[17] Gamma, Erich; Richard Helm, Ralph Johnson, John M. Vlissides; *Design Patterns: Elements of Reusable Object-Oriented Software*; Addison-Wesley, Dátum aktualizácie/revízie: 1995, [cit. marec 2008]; ISBN 0201633612; 395 str.

[18] John M. Boyer, IBM; *XForms 1.0 (Third Edition) W3C Recommendation;* [online]. Dátum aktualizácie/revízie: október 2007, [cit. apríl 2008]. Dostupné na internete: <http://www.w3.org/TR/xforms/>

[19] *Adobe XML Forms Architecture (XFA),* domovská stránka technológie [online]. Dátum aktualizácie/revízie: január 2008, [cit. apríl 2008]. Dostupné na internete: <http://partners.adobe.com/public/developer/xml/index_arch.html>

Príloha

V prílohe je obsah wsdl súboru humanTasks.wsdl spolu so súvisiacou XML schémou humanTaskTypes.xsd. Tieto súbory spoločne popisujú schému pre zápis aplikovaných vzorov a s nimi súvisiace atribúty.

humanTasks.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="humanTasks"
    targetNamespace="http://humanTask.example.org/"
    xmlns="http://humanTask.example.org/"
    xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    xmlns:ht="http://xml.uniba.sk/ht"
    xmlns:plk="http://docs.oasis-open.org/wsbpel/2.0/plnktype"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:sonicxq="http://www.sonicsw.com/sonicxq/"
    xmlns:tns="http://humanTask.example.org/"
    xmlns:vprop="http://docs.oasis-open.org/wsbpel/2.0/varprop"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <plk:partnerLinkType name="DiplomovkaPortType_PLT">
        <plk:role name="DiplomovkaPortType_MsgsReceiver"
            portType="tns:humanTasks"/>
    </plk:partnerLinkType>

    <wsdl:types>
        <xsd:schema targetNamespace="http://humanTask.example.org/">
            <xsd:import namespace="http://xml.uniba.sk/ht"
                  schemaLocation="humanTaskTypes.xsd"/>

            <xsd:element name="handleHumanTask">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="header" type="ht:header"
                              minOccurs="0"/>
                        <xsd:element name="Data" type="ht:Data"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

            <xsd:element name="HumanTaskResponse">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="UserResponse"
                              type="ht:UserResponse"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>

        </xsd:schema>
    </wsdl:types>

    <wsdl:message name="handleHumanTaskRequest">
        <wsdl:part element="tns:handleHumanTask" name="parameters"/>
    </wsdl:message>
```

```
    <!-- Popis odpovede od ľudskej úlohy po jej vykonaní pre vykonávacie
      prostredie BPEL -->
    <wsdl:message name="HumanTaskResponse">
        <wsdl:part element="tns:HumanTaskResponse" name="parameters"/>
    </wsdl:message>

    <wsdl:portType name="humanTasks">
        <wsdl:operation name="handleHumanTask">
            <wsdl:input message="tns:handleHumanTaskRequest"/>
            <wsdl:output message="tns:HumanTaskResponse"/>
        </wsdl:operation>
    </wsdl:portType>

    <wsdl:binding name="humanTasksSOAP" type="tns:humanTasks">
        <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="handleHumanTask">
            <soap:operation soapAction="" style="document"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="humanTasksService">
        <wsdl:port binding="tns:humanTasksSOAP" name="humanTasksSOAP">
            <soap:address location="http://localhost:9000/HT"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

humanTaskTypes.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema elementFormDefault="qualified"
    targetNamespace="http://xml.uniba.sk/ht"
    xmlns="http://xml.uniba.sk/ht"
    xmlns:local="http://xml.uniba.sk/ht"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <!-- Údaje pre TM resp. TLB popisujúce aplikované vzory na úlohu a ich
      parametre -->
    <xsd:complexType name="header">
        <xsd:sequence>
            <!-- Zoznam vzorov aplikovaných na úlohu -->
            <xsd:element maxOccurs="unbounded" minOccurs="1" name="patterns">
                <xsd:complexType>
                    <xsd:choice>
                        <xsd:element name="DirectDistribution"
                              type="local:DirectDistribution"/>
                        <xsd:element name="RoleBasedDistribution"
                              type="local:RoleBasedDistribution"/>
                        <xsd:element name="RetainFamiliar"
                              type="local:RetainFamiliar"/>
                        <xsd:element name="CaseHandling"
                              type="local:CaseHandling"/>
```

```
                    <xsd:element name="SeparationOfDuties"
                          type="local:SeparationOfDuties"/>
                    <xsd:element name="CapabilityBasedDistribution"
                          type="local:CapabilityBasedDistribution"/>
                </xsd:choice>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>

<!-- Popis zdroja -->
<xsd:complexType name="resource">
    <xsd:all>
        <xsd:element name="ID" type="xsd:positiveInteger"/>
        <xsd:element minOccurs="0" name="ResourceName"
              type="xsd:positiveInteger"/>
    </xsd:all>
</xsd:complexType>

<!-- Definícia roly pomocou zoznamu zdrojov -->
<xsd:complexType name="Role">
    <xsd:sequence>
        <xsd:element name="RoleName" type="xsd:string"/>
        <xsd:element maxOccurs="unbounded" name="Resources"
              type="local:resource"/>
    </xsd:sequence>
</xsd:complexType>

<!-- Zápis schopnosti pre podporu vzoru 8) Capability-Based Allocation -->
<xsd:complexType name="Capability">
    <xsd:all>
        <xsd:element name="CapabilityName" type="xsd:string"/>
        <xsd:element name="CapabilityType" type="xsd:string"/>
        <xsd:element name="CapabilityValue" type="xsd:string"/>
    </xsd:all>
</xsd:complexType>

<!-- Popisy jednotlivých vzorov spolu s potrebnými atribútmi a pod. -->
<xsd:complexType name="Pattern">
    <xsd:attribute name="ID" type="xsd:integer"/>
</xsd:complexType>

<xsd:complexType name="DirectDistribution">
    <xsd:complexContent>
        <xsd:extension base="local:Pattern">
            <xsd:all>
                <xsd:element name="Resource" type="local:resource"/>
            </xsd:all>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="RoleBasedDistribution">
    <xsd:complexContent>
        <xsd:extension base="local:Pattern">
            <xsd:sequence>
                <xsd:element maxOccurs="unbounded" name="Roles"
                    type="local:Role"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
```

```
</xsd:complexType>

<xsd:complexType name="RetainFamiliar">
    <xsd:complexContent>
        <xsd:extension base="local:Pattern">
            <!-- Každá skupina úloh spĺňajúca tento vzor má ID. -->
            <xsd:attribute name="groupID" type="xsd:integer"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- Pri vzore 6) Case Handling je iba jedna skupina. -->
<xsd:complexType name="CaseHandling">
    <xsd:complexContent>
        <xsd:extension base="local:Pattern"/>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="SeparationOfDuties">
    <xsd:complexContent>
        <xsd:extension base="local:Pattern">
              <!-- Každá skupina úloh spĺňajúca tento vzor má ID. -->
            <xsd:attribute name="groupID" type="xsd:integer"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="CapabilityBasedDistribution">
    <xsd:complexContent>
        <xsd:extension base="local:Pattern">
            <xsd:sequence>
                <xsd:element maxOccurs="unbounded"
                    name="Capabilities" type="local:Capability"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- Čisté aplikačne dáta - napr. XML formulár -->
<xsd:complexType name="Data"/>

<xsd:complexType name="TextData">
    <xsd:complexContent>
        <xsd:extension base="local:Data">
            <xsd:sequence>
                <xsd:element name="Text" type="xsd:string"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="BinaryData">
    <xsd:complexContent>
        <xsd:extension base="local:Data">
            <xsd:sequence>
                <xsd:element name="data" type="xsd:hexBinary"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

```
    <!-- Odpoveď po vykonaní ľudskej úlohy pre vykonávacie prostredie BPEL -->
    <xsd:complexType name="UserResponse">
        <xsd:sequence>
            <xsd:element name="Approved" type="xsd:boolean" minOccurs="0"/>
            <xsd:element name="Data" type="local:Data" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```