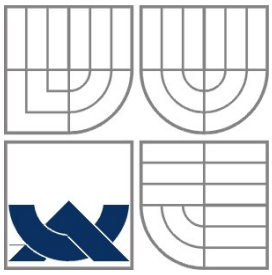
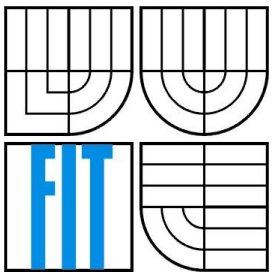


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉM PRO PODPORU ŘÍZENÍ WORKFLOW

SYSTEM FOR WORKFLOW MANAGEMENT SUPPORT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Tomáš Slobodník
AUTHOR

VEDOUCÍ PRÁCE — Ing. Šárka Květoňová, Ph.D.
SUPERVISOR

BRNO 2009

Abstrakt

Práce se věnuje návrhu a implementaci služby procesního řízení a pojednává o jazyce XPDL, sloužícímu k uchování a přenosu procesních definic, na kterém je tato služba založena. Text obsahuje praktickou ukázku funkce.

Abstract

Paper describes design and implementation of process management service and discuss XPDL language, which is used for storing and exchanging processes definitions. Text contains practical example.

Klíčová slova

XPDL, workflow, BPM, procesní řízení

Keywords

XPDL, workflow, BPM, process management

Citace

SLOBODNÍK, Tomáš: *Systém pro podporu řízení workflow*, bakalářská práce, Brno, FIT VUT v Brně, 2009

Systém pro podporu řízení workflow

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Šárky Květoňové, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Tomáš Slobodník
15. května 2009

Poděkování

Rád bych poděkoval vedoucí práce, Šárce Květoňové, za odbornou a stylistickou pomoc při jejím psaní.

Obsah

1 Úvod

1.1 Motivace

Žijeme v rychle se rozvíjejícím světě. Lidská společnost, a posledních několik desetiletí i technika, se vyvíjí rychleji než kdykoliv předtím. V takto dynamickém světě je nesmírně důležité, aby firmy a společnosti byly schopné operativně reagovat na změny trhu a patřičnou rychlostí se rozpínat. Předpokladem pro úspěšný rozvoj moderní společnosti je mimo jiné i perfektně zvládnutý Business Process Management. Při jeho nezvládnutí by se rychle rostoucí společnost mohla snadno proměnit v chaos a její růst by tím byl zastaven, ba dokonce by se mohla do ohrožení dostat samotná její existence.

Proto společnosti stále více využívají komplexních technických prostředků, v podobě nejrůznějších informačních systémů, které jim pomáhají udržet nad společností a jejím rozvojem kontrolu.

1.2 Systém pro podporu řízení workflow

Jedním z těchto podpůrných systémů je systém pro řízení procesů uvnitř společnosti. Ten sestává z několika částí. V prvé řadě je třeba nástroj pro modelování procesů, dnes nejčastěji grafický, kde modelování probíhá formou kreslení diagramu vývoje procesu a nastavováním příslušných vlastností a dalších potřebných informací k jednotlivým prvkům. Výstupem takového nástroje pro modelování – designéru, je definice procesu v některém standardizovaném či vlastním jazyce. Na internetu lze najít i nějaké freeware či dokonce open-source nástroje, ale vzhledem k tomu, že ani standardizované jazyky pro definici procesu nedefinují způsob ukládání všech nezbytných částí procesu, je toto ponecháno na tvůrcích designéru, což je dělá neuniverzálními či přímo použitelnými jen s odpovídajícími dalšími nástroji.

Dalším prvkem je samotný systém pro provádění a řízení procesů podle předložené definice. Jde vlastně o serverovou část systému – službu, která kromě výkonného jádra komunikuje s klienty a pomocí vzájemné interakce proces vykonává.

Klientská část systému slouží pro předávání informací mezi uživatelem a výkonnou částí systému. Nejsnáze si lze klienta představit jako program, který po síti komunikuje se serverem a výsledky komunikace zobrazuje uživateli v grafickém rozhranní.

1.3 Stanovené cíle

Cílem této práce je návrh a implementace právě serverové části systému, která bude sloužit k řízení procesů uvnitř společnosti – systém pro podporu workflow. Systém by měl splňovat požadavky na moderní řešení – měl by být spolehlivý, rychlý a mít rozsáhlé možnosti nastavení pro splnění každého požadavku.

Aby byl takový systém pro moderní společnost opravdovým přínosem, musí umožňovat procesy dynamicky měnit a přizpůsobovat aktuálním požadavkům.

Tato práce navazuje na bakalářskou práci Zdeňka Soukupa, jejímž výstupem je designér pro modelování procesů.

2 XML Process Definition Language

Zkráceně XPDL, je jazyk sloužící k uchování definice procesu v textové podobě ve formátu XML. Jazyk dnes existuje v několika verzích a je standardizovaný Workflow Management Coalition (WfMC). V této práci je použita verze XPDL 1.0, která byla uvedena roku 2002[1] a slouží jako náhrada za Workflow Process Definition Language (WPDL)[2], kdy hlavním rozdílem a nespornou výhodou proti tomuto staršímu standardu je použití formátu XML, který je dnes velmi rozšířený a podporovaný.

2.1 Definice XPDL

XPDL je XML formát umožňující definovat procesy a slouží k výměně informací mezi modelovacím nástrojem a službou pro jejich provádění.[3] Stejně tak může sloužit k přesunu mezi modelovacími nástroji. XPDL dokument obsahuje jednu nebo více procesních definic, z nichž každá je reprezentována množinou aktivit (dílčích úkolů v rámci procesu) a cest mezi nimi.

Dokument je orientován do grafu, kde jednotlivé aktivity jsou reprezentovány jeho uzly. K zobrazení v grafické podobě lze využít např. notaci BPMN (Bussines Process Modeling Notation), která je prakticky vyvíjena pro kooperaci s XPDL a jsou tedy navzájem plně kompatibilní. Řada designérů s možností exportu do XPDL je právě na této notaci založena.

Specifikace dokumentu obsahuje několik základních elementů, jak znázorňuje Obr. 1[1]. Každý jednotlivý soubor XPDL dokumentu obsahuje v kořenu právě jeden element `Package`, ve kterém je povinný element `PackageHeader` a žádný nebo jeden výskyt každého z elementů, jak je znázorňuje specifikace elementu `Package`:

```
<xsd:element name="Package">
    <xsd:complexType>
       <xsd:sequence>
           <xsd:element ref="xpdl:PackageHeader"/>
```

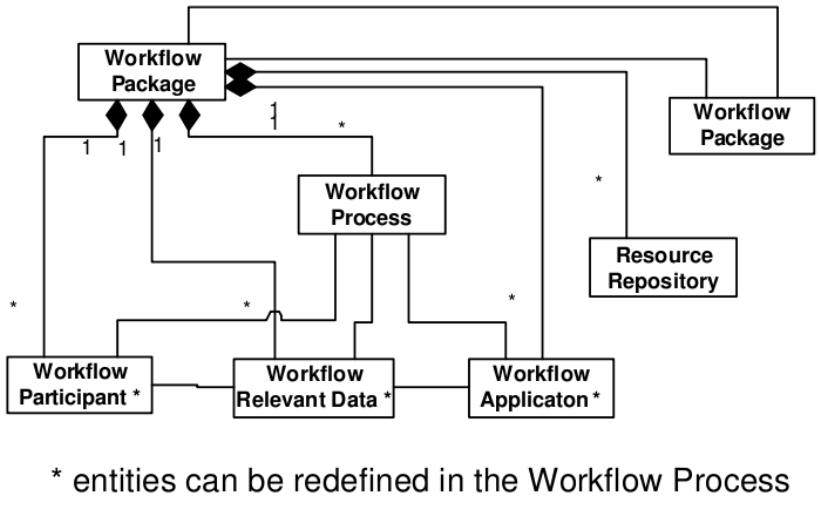
1 Workflow Management Coalition Workflow Standard : *Workflow Process Definition Interface – XML Process Definition Language* [online]. Workflow Management Coalition, 2002, October 25, 2002 [cit. 2009-04-24]. Dostupný z WWW: <http://xml.coverpages.org/XPDLv10.pdf>.

2 ROZENBERG, Grzegorz. Advanced Course on Petri Nets. [s.l.] : Springer, 2004. 849 s. ISBN 9783540222613. Str. 39.

3 HAVEY, Michael. *Essential business process modeling*. [s.l.] : O\'Reilly, 2005. 332 s. ISBN 9780596008437. Str. 178.

```
            <xsd:element ref="xpdl:RedefinableHeader" minOccurs="0"/>
            <xsd:element ref="xpdl:ConformanceClass" minOccurs="0"/>
            <xsd:element ref="xpdl:Script" minOccurs="0"/>
            <xsd:element ref="xpdl:ExternalPackages" minOccurs="0"/>
            <xsd:element ref="xpdl:TypeDeclarations" minOccurs="0"/>
            <xsd:element ref="xpdl:Participants" minOccurs="0"/>
            <xsd:element ref="xpdl:Applications" minOccurs="0"/>
            <xsd:element ref="xpdl:DataFields" minOccurs="0"/>
            <xsd:element ref="xpdl:WorkflowProcesses" minOccurs="0"/>
            <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
        <xsd:attribute name="Name" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>
```

Díky elementu `ExternalPackages` jsme současně schopni odkazovat na jiné XPDL soubory, což nám umožňuje logické rozdělení dokumentu podle jednotlivých elementů, množin souvisejících procesů, apod. Tímto výrazně podpoříme přehlednost zejména v rozsáhlých dokumentech.



Obr. 1: Package meta model

Samotný workflow proces je definován uvnitř elementu `WorkflowProcess`, který je členěn následovně:

```
<xsd:element name="WorkflowProcess">
   <xsd:complexType>
       <xsd:sequence>
          <xsd:element ref="xpdl:ProcessHeader"/>
          <xsd:element ref="xpdl:RedefinableHeader" minOccurs="0"/>
          <xsd:element ref="xpdl:FormalParameters" minOccurs="0"/>
          <xsd:group ref="xpdl:DataTypes"/>
          <xsd:element ref="xpdl:DataFields" minOccurs="0"/>
          <xsd:element ref="xpdl:Participants" minOccurs="0"/>
          <xsd:element ref="xpdl:Applications" minOccurs="0"/>
          <xsd:element ref="xpdl:ActivitySets" minOccurs="0"/>
          <xsd:element ref="xpdl:Activities" minOccurs="0"/>
```

```
            <xsd:element ref="xpdl:Transitions" minOccurs="0"/>
            <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
        <xsd:attribute name="Name" type="xsd:string"/>
        <xsd:attribute name="AccessLevel">
            <xsd:simpleType>
               <xsd:restriction base="xsd:NMTOKEN">
                   <xsd:enumeration value="PUBLIC"/>
                   <xsd:enumeration value="PRIVATE"/>
               </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
</xsd:element>
```

2.1.1 Atribut

Definice atributů, se kterými proces pracuje, se nachází uvnitř elementu `DataFields`. K dispozici jsou všechny základní datové typy jako INTEGER, STRING, FLOAT. Je ale možné si pomocí elementu `DeclaredType` definovat i vlastní typy. Toto nachází uplatnění především pro definici různých výčtových typů, kdy potřebujeme sledovat stav nebo přibližnou hodnotu definovanou intervalem.

Atributy lze definovat přímo v elementu konkrétního workflow procesu a slouží tak pouze jako lokální atributy pro daný proces. Druhá možnost je definovat atributy přímo v elementu `Package` a takto definované jsou potom dostupné pro všechny procesy.

```
<xsd:element name="DataField">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="xpdl:DataType"/>
            <xsd:element ref="xpdl:InitialValue" minOccurs="0"/>
            <xsd:element ref="xpdl:Length" minOccurs="0"/>
            <xsd:element ref="xpdl:Description" minOccurs="0"/>
            <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
        <xsd:attribute name="Name" type="xsd:string"/>
        <xsd:attribute name="IsArray" default="FALSE">
            <xsd:simpleType>
               <xsd:restriction base="xsd:NMTOKEN">
                   <xsd:enumeration value="TRUE"/>
                   <xsd:enumeration value="FALSE"/>
               </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
    </xsd:complexType>
</xsd:element>

<xsd:group name="DataTypes">
        <xsd:choice>
               <xsd:element ref="xpdl:BasicType"/>
               <xsd:element ref="xpdl:DeclaredType"/>
               <xsd:element ref="xpdl:SchemaType"/>
               <xsd:element ref="xpdl:ExternalReference"/>
               <xsd:element ref="xpdl:RecordType"/>
               <xsd:element ref="xpdl:UnionType"/>
               <xsd:element ref="xpdl:EnumerationType"/>
               <xsd:element ref="xpdl:ArrayType"/>
               <xsd:element ref="xpdl:ListType"/>
```

```
        </xsd:choice>
</xsd:group>
```

2.1.2 Krok (Aktivity)

Kroky neboli aktivity jsou dílčí činnosti, ze kterých proces sestává. XPDL obsahuje tyto čtyři typy kroků: normální (atomický), subflow, blokový a směrovací (route).

Normální krok je základním stavebním kamenem procesu. Obsahuje množinu úkolů vykonávaných určitou rolí (viz. kapitola 2.1.3), která je přiřazena elementem `Performer`. Ke každému kroku může být přiřazena právě jedna role.

Subflow je typ kroku, který spustí další workflow proces. Nemá přístup k atributům, se kterými pracuje proces, který ho vyvolal, ale data si mohou předat pomocí vstupních a výstupních parametrů subflow kroku. Subflow může běžet asynchronně s procesem, ze kterého byl vyvolán.

Blokový krok obsahuje množinu kroků propojených cestami. Na rozdíl od subflow má přístup ke všem datům, se kterými proces pracuje. Je to v podstatě abstrakce nad určitou částí procesu.

Směrovací krok je prázdný krok, který neobsahuje žádné úkoly ke zpracování a je řízen systémem. Účelem kroku je rozšíření možností rozhodování o směru toku kaskádováním podmínek v cestách. Typickým příkladem pro použití směrovacího kroku je vytvoření konstrukce *if-elseif -else*.

```
<xsd:element name="Activity">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element ref="xpdl:Description" minOccurs="0"/>
                        <xsd:element ref="xpdl:Limit" minOccurs="0"/>
                        <xsd:choice>
                                <xsd:element ref="xpdl:Route"/>
                                <xsd:element ref="xpdl:Implementation"/>
                                <xsd:element ref="xpdl:BlockActivity"/>
                        </xsd:choice>
                        <xsd:element ref="xpdl:Performer" minOccurs="0"/>
                        <xsd:element ref="xpdl:StartMode" minOccurs="0"/>
                        <xsd:element ref="xpdl:FinishMode" minOccurs="0"/>
                        <xsd:element ref="xpdl:Priority" minOccurs="0"/>
                        <xsd:element ref="xpdl:Deadline" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element ref="xpdl:SimulationInformation" minOccurs="0"/>
                        <xsd:element ref="xpdl:Icon" minOccurs="0"/>
                        <xsd:element ref="xpdl:Documentation" minOccurs="0"/>
                        <xsd:element ref="xpdl:TransitionRestrictions" minOccurs="0"/>
                        <xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
                </xsd:sequence>
                <xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
                <xsd:attribute name="Name" type="xsd:string"/>
        </xsd:complexType>
</xsd:element>
```

2.1.3 Role (Participant)

Role slouží jako abstrakce mezi kroky a uživateli, kteří je vykonávají. Nejběžnějšími typy rolí jsou: *role*, *human*, *organizational unit*, *system*. Rolím lze přiřadit konkrétní vykonavatele, ať už jsou to lidé či programy.

Role typu *role* slučuje uživatele vykonávající určitou stálou funkci. Např. technik nebo obchodník. Role *human* představuje ve společnosti nějaké unikátní postavení zastoupené právě jednou osobou, např. vedoucí výroby. *Organizational unit* poté slouží ke slučování množiny rolí jiného typu a systém je role, kterou zastupuje samotný systém.

```
<xsd:element name="Participant">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element ref="xpdl:ParticipantType"/>
			<xsd:element ref="xpdl:Description" minOccurs="0"/>
			<xsd:element ref="xpdl:ExternalReference" minOccurs="0"/>
			<xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
		</xsd:sequence>
		<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
		<xsd:attribute name="Name" type="xsd:string"/>
	</xsd:complexType>
</xsd:element>
```

2.1.4 Cesta (Transition)

Množina kroků v procesu je propojena pomocí cest, které určují směr toku v procesu a současně definují i podmínky přechodů z kroku do kroku a řeší větvení toku. Každá cesta obsahuje tři základní informace: z jakého kroku vychází, do kterého kroku vede a podmínku, jejímž splněním bude přechod proveden.

```
<xsd:element name="Transition">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element ref="xpdl:Condition" minOccurs="0"/>
			<xsd:element ref="xpdl:Description" minOccurs="0"/>
			<xsd:element ref="xpdl:ExtendedAttributes" minOccurs="0"/>
		</xsd:sequence>
		<xsd:attribute name="Id" type="xsd:NMTOKEN" use="required"/>
		<xsd:attribute name="From" type="xsd:NMTOKEN" use="required"/>
		<xsd:attribute name="To" type="xsd:NMTOKEN" use="required"/>
		<xsd:attribute name="Name" type="xsd:string"/>
	</xsd:complexType>
</xsd:element>
```

Podmínka, kterou cesta obsahuje, může mít několik typů: *conditional*, *otherwise*, *exception* a *default exception*. Cesta také nemusí mít podmínku žádnou je tedy provedena vždy. Podmínky *conditional* a *otherwise* si lze nejsnáze představit na konstrukci if-else, kdy pokud není splněna podmínka v cestě typu *conditional* (if), bude vykonána cesta typu *otherwise* (else). Typům *exception*

a *default exception* se v této práci věnovat nebudeme, protože XPDL je v použité verzi nedefinuje. Specifikace podmínky je následující:

```
<xsd:element name="Condition">
	<xsd:complexType mixed="true">
		<xsd:choice minOccurs="0" maxOccurs="unbounded">
			<xsd:element ref="xpdl:Xpression"/>
		</xsd:choice>
		<xsd:attribute name="Type">
			<xsd:simpleType>
				<xsd:restriction base="xsd:NMTOKEN">
					<xsd:enumeration value="CONDITION"/>
					<xsd:enumeration value="OTHERWISE"/>
					<xsd:enumeration value="EXCEPTION"/>
					<xsd:enumeration value="DEFAULTEXCEPTION"/>
				</xsd:restriction>
			</xsd:simpleType>
		</xsd:attribute>
	</xsd:complexType>
</xsd:element>
<xsd:element name="Xpression">
	<xsd:complexType mixed="true">
		<xsd:choice minOccurs="0" maxOccurs="unbounded">
			<xsd:any processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
		</xsd:choice>
	</xsd:complexType>
</xsd:element>
```

2.2 Vlastní interpretace

Jazyk XPDL sám o sobě nedefinuje veškeré prvky procesu nezbytné k jeho provedení. Zásadní je zde především pohyb v grafu procesu, který se řídí vyhodnocováním podmínek v cestách. Pokud má být XPDL výstup z designéru spustitelný na serverové části, je třeba definovat jistý skriptovací jazyk, v jehož syntaxi budou podmínky zapsány a který bude server schopen interpretovat a vyhodnotit.

Základní dovedností bude schopnost porovnávat hodnoty atributů s konstantami, jinými atributy, či symboly se speciálním významem. K porovnávání využijeme standardních operátorů z programovacích jazyků: `=, !=, >, <,` tedy rovnost, nerovnost, větší než a menší než.

Protože podmínky se v modelovacím nástroji budou psát ručně, je vhodné, aby se názvy atributů zapisovaly ve vizuálně srozumitelné formě, tedy nejlépe podle jejich názvů. Takový název ovšem může být víceslovný a proto bude celý název atributu v podmínkovém výrazu uzavřen mezi znaky `[]`. Toto výrazně zjednoduší rozdělování výrazu na tokeny při interpretaci. Takový zápis tedy bude vypadat např. takto:

```
[Stav skladu] < 1000
```

Tímto už jsme schopni porovnávat celočíselné hodnoty, hodnoty v plovoucí řádové čárce, stejně jako dva atributy číselného typu mezi sebou.

Pro porovnání atributu typu string (textový řetězec) s konstantou, uzavřeme tento do uvozovek. Můžeme tak provést i takovéto porovnání:

```
[Adresa] = "Pod Kaštany 42, Brno"
```

O použitelnosti takového výrazu se dá jistě úspěšně pochybovat, ale v jiném případě, než na porovnávání adresy, své uplatnění najde. V praxi však bude frekventovanější kontrola, zda atribut typu řetězec vůbec nějakou hodnotu obsahuje. To můžeme realizovat pomocí zápisu pravé strany výrazu ve tvaru `""` nebo, pro větší přehlednost, použít speciálního symbolu `NULL`, který představuje prázdnou hodnotu.

Jelikož v praxi potřebujeme často vyhodnocovat podle hodnoty více než jednoho atributu, tak aby nemusela definice procesu obsahovat zbytečně velké množství směrovacích kroků (viz. kapitola 2.1.2), zavedeme operátory konjunkce a disjunkce `AND`, `OR`. Díky nim můžeme rozhodnutí provést na základě výsledků většího počtu atomických výrazů.

V případě potřeby složitější podmínky na cestě bude nevyhnutelné využít některý plnohodnotný skriptovací jazyk. Skript se v takovém případě bude nahrávat na server spolu s XPDL specifikací, v podmínce cesty bude uvedena jen reference na něj a vstupní parametry. Server v takové situaci skript spustí v odpovídajícím interpreteru, předá mu potřebné hodnoty atributů procesu a po dokončení převezme návratovou hodnotu – `true` nebo `false`.

3 Návrh systému

Jádrem celého systému je zcela jistě jeho serverová část, která musí bezchybně spolupracovat se všemi ostatními prvky, tedy s designérem procesů, klientskými aplikacemi a nástrojem pro správu. Jelikož však žádný program není bezchybný a rozsah činností, které musí server vykonávat, není malý, je nasnadě provést návrh dostatečně modulárně, aby bylo možné co nejsnáze upravovat jeho chování a přidávat nové vlastnosti bez nutnosti zasahovat do základního konceptu. Také volba dostatečně spolehlivé a rozšířené platformy, na které bude server vyvíjen, přispěje k zajištění dobré použitelnosti v praxi.

První a jistě nejdůležitější krok je návrh serverové databáze. Tato uchovává veškerá perzistentní data potřebná pro běh systému, stejně jako historii prováděných procesů, informace o uživatelích, opravněních, apod. Databáze musí umožňovat uchovávat veškerá data a reference, která může server vyžadovat, a to i ta, jejichž obsluha zatím není implementována, ale počítá se s nimi při budoucím vývoji. Cílem je zabránit situaci, kdy je nutné provést zásadnější změny v již naplněné databázi, což by vedlo k nepříjemné nutnosti veškerá existující data ručně převést do nové databáze nebo se smířit s jejich ztrátou. To však většinou nepřipadá v úvahu. Pravděpodobně by takový zásah vyžadoval i nežádoucí delší odstavení systému.

Správa serveru bude zajišťována síťovou komunikací s administračním nástrojem, což zajistí snadnou dostupnost z libovolného místa v síti, bez nutnosti připojovat se přímo na fyzický stroj, kde služba běží. Administrátor tohoto systému také nemusí mít na fyzický stroj žádný jiný přístup, proto by pomocí tohoto nástroje měla být umožněna kompletní správa serveru, včetně např. nahrávání nové definice procesů či restart serveru. Samotný server nebude obsahovat žádné grafické rozhranní, toto bude existovat jako samostatná aplikace komunikující pomocí socketů. Tímto fyzickým oddělením dojde ke zjednodušení jádra serveru a současně to umožní provádět změny nástroje pro správu bez nutnosti zasahovat do činnosti serveru.

Předávání informací mezi serverem a uživateli bude probíhat taktéž po síti. Protokol, kterým bude tato komunikace realizována, musí mít dostatečné možnosti, aby se klientská aplikace dala napsat dostatečně uživatelsky přívětivě a přitom byla zachována efektivita síťové komunikace. Pevně definovaný protokol umožní také vytvoření různých klientských aplikací pro různé použití a platformy.

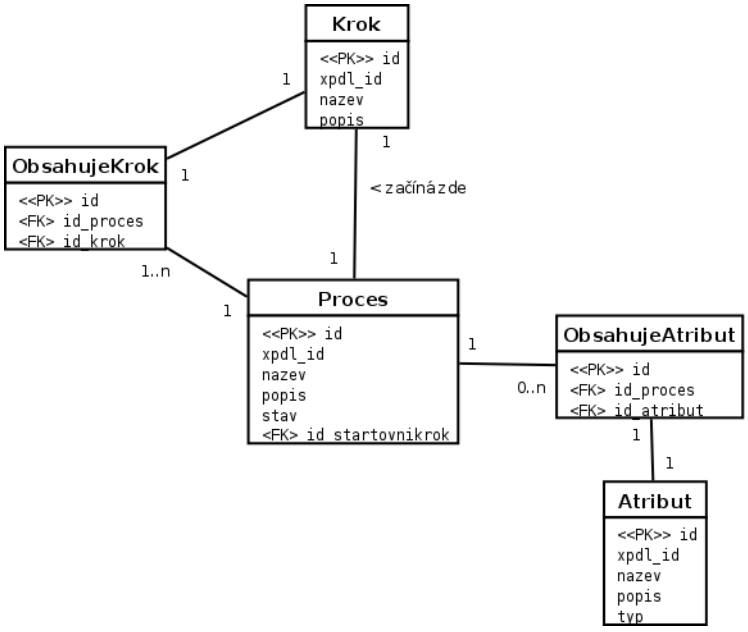
Zvláštní důraz je také třeba věnovat periferním částem, které komunikují s okolím a to převážně komunikaci s klientskými aplikacemi a administračním rozhranním, protože změny v této komunikaci by zapříčinili nutnost úprav i těchto aplikací.

3.1 Databáze

Databáze slouží k udržování dat systému a pro znovuobnovení činnosti v případě odstavení či poruchy. Proto budou do databáze načtena prakticky všechna data z definičních souborů procesů.

Základní data pro uložení do databáze jsou informace o procesech, jejichž definici máme k dispozici. Tato metadata jsou obsažena v jedné entitě a obsahují název, popis a identifikátor použitý v XPDL dokumentu, ze kterého jsou také všechny informace načteny. Dále je zde stav, určující, zda je proces aktivní, tedy zda je možné ho založit, nebo neaktivní, kdy ho uživatelé vůbec neuvidí v nabídce dostupných procesů.

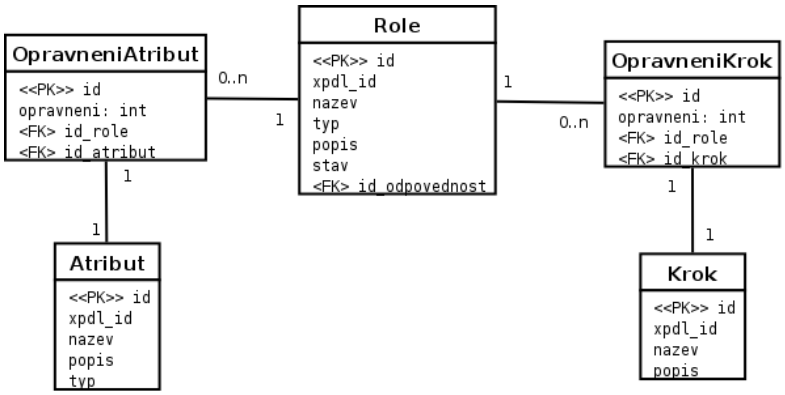
Další nezbytnou součástí procesu jsou jeho jednotlivé kroky a atributy, se kterými pracuje. V obou těchto entitách najdeme stejné informace, jako v případě procesu, jen u atributu je navíc typ, který určuje datový typ atributu, a poznáme podle něj, jak s daným atributem pracovat. Včetně nezbytných entit, zajišťujících správné přiřazení k procesům, je tato část znázorněna na Obr. 2



Obr. 2: ER diagram - proces, atribut, krok

Nyní v XPDL dokumentu zbývají již jen dva elementy obsahující informace důležité pro běh procesu – `Transitions` a `Participants`. `Transitions`, tedy cesty, nepotřebujeme mít uložené v databázi, protože s nimi vzhledem k jejich statičnosti nebudou tvořit vztahy žádné další entity. Zbývá tedy už uchovat pouze jednotlivé role a jejich vztahy k atributům a procesům. Jelikož do rolí se přiřazují uživatelé a jejich činnost musíme být schopni, z důvodu bezpečnosti a správného průběhu procesu, řídit, jsou role ideální místo na nastavování oprávnění. Ta se budou nastavovat v

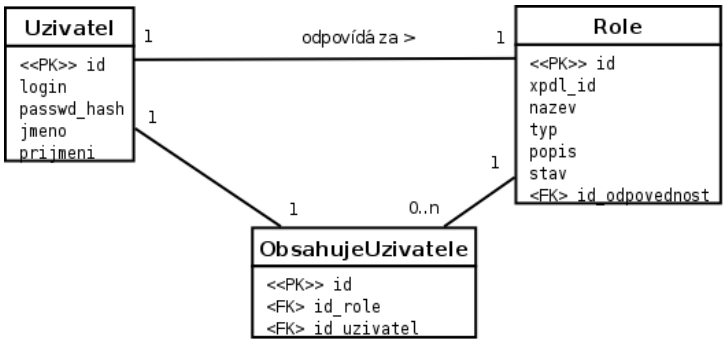
entitách určujících příslušnost rolí ke krokům a atributům. Položka určující oprávnění bude pro jednoduchost typu integer, tedy celočíselná hodnota a bude nabývat tří hodnot: 0 – žádná oprávnění, 1 – oprávnění ke čtení, 2 – oprávnění ke čtení i zápisu.



Obr. 3: ER diagram - Role, oprávnění

V případě neexistence entity definující oprávnění mezi dvojicí role-atribut nebo role-krok bude toto bráno jako ekvivalent oprávnění 0. Taková situace může snadno nastat, protože automaticky budou nastavována jen ta oprávnění, která se načtou z XPDL dokumentu. Ty definují nezbytná práva pro běh procesu a zpřístupní kroky a atributy těm rolím, které s nimi musí pracovat (v XPDL jsou nastaveny jako `Performers`). Zbytek oprávnění se definuje prostřednictvím administračního nástroje.

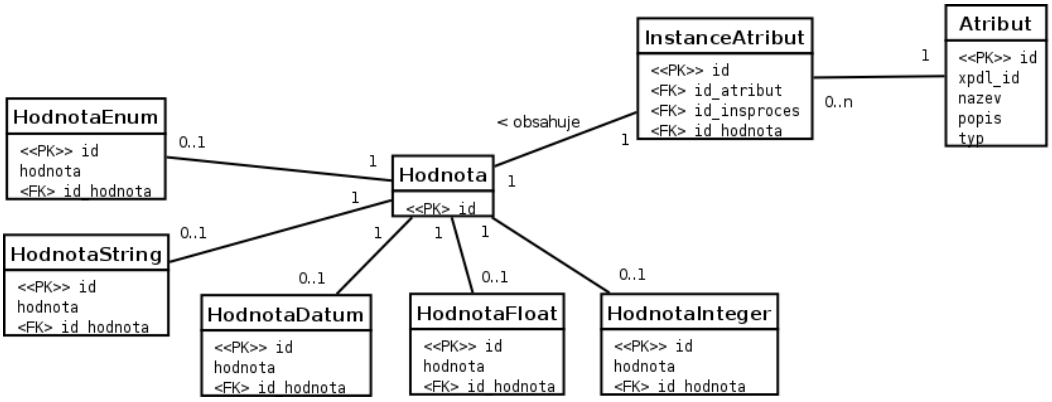
Jelikož role slouží pouze jako abstrakce nad libovolnou výkonnou jednotkou, je třeba k nim přiřadit skutečné uživatele, kteří budou roli zastupovat. O uživateli je v databázi uložen login, unikátní řetězec, který uživatel používá při autentizaci, hash řetězec z autentizačního hesla a dále jméno a přijmení.



Obr. 4: ER diagram - Vztah role-uživatel

Každé roli je také přiřazena odpovědná osoba (`<FK> id_odpovednost`), která v systému vystupuje jako běžný uživatel, ale současně má speciální oprávnění, vztahující se k roli, jejíž je odpovědnou osobou. Toto jí umožňuje např. upravit hodnotu atributu, který byl nastaven uživatelem této role. V této práci nejsou řešeny speciální „nelidské“ role, s výjimkou role SYSTEM.

V tuto chvíli můžeme uložit do databáze veškerá data, která jsou v rámci běhu procesu neměnná. Aby však data běžících procesů byla také perzistentní pro případ ukončení a opětovného nastartování serveru z libovolného důvodu, potřebujeme pro každý proces zvlášť uložit všechna relevantní data. Na Obr. 5 je vidět, jak toto řešení vypadá v případě instancí atributů.



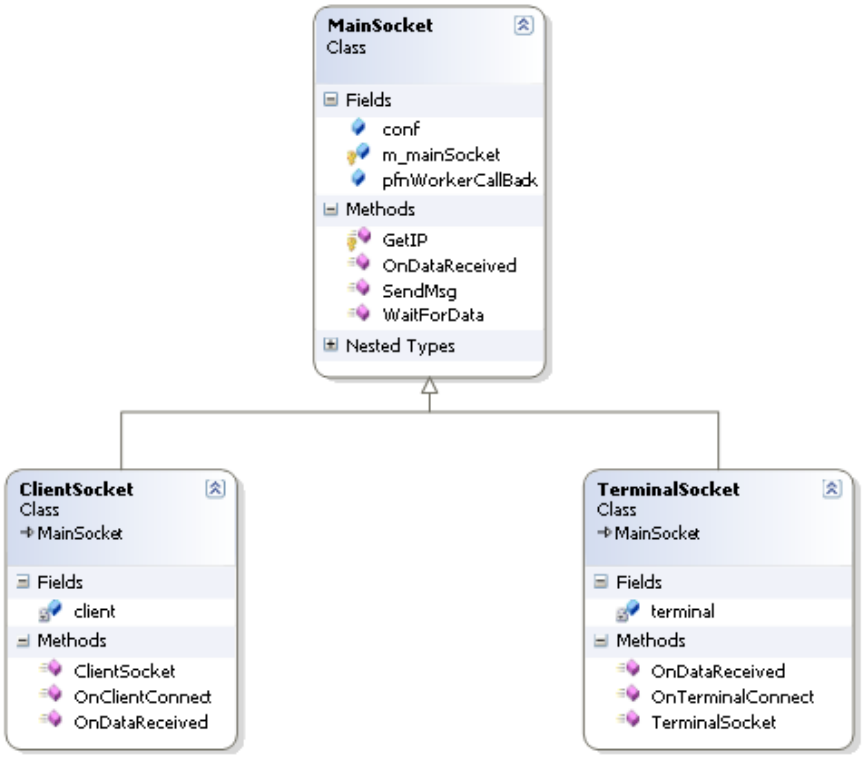
Obr. 5: ER diagram - Instance atributu

Atributy jsou v tomto poněkud speciální z toho důvodu, že obecně mohou mít několik různých datových typů. Vždy se použije jen jeden datový typ podle typu konkrétního atributu. Při použití sloupců všech datových typů v jedné tabulce by docházelo ke zbytečné redundanci, a současně by to komplikovalo databázové dotazy. Proto je zavedena entita `Hodnota`, která obsahuje pouze primární klíč a poté množina entit odpovídající počtu datových typů, kdy každá tato entita obsahuje pouze sloupec toho datového typu, který představuje a cizí klíč na entitu `Hodnota`. `InstanceAtribut` poté obsahuje ten samý cizí klíč, čímž je vytvořena reference. Entita `Hodnota` tedy slouží jako meta entita.

Kompletní ER diagram viz. Příloha 1.

3.2 Síťové řešení

Jelikož komunikace po síti je jediné spojení serveru s okolím, je důležité, aby síťová část obsahovala dostatečné možnosti řízení činnosti serveru, a současně aby tato komunikace byla spolehlivá. Proto budou datové přenosy po síti realizovány pomocí TCP/IP, které garantuje správnost doručených paketů a jejich správné pořadí.



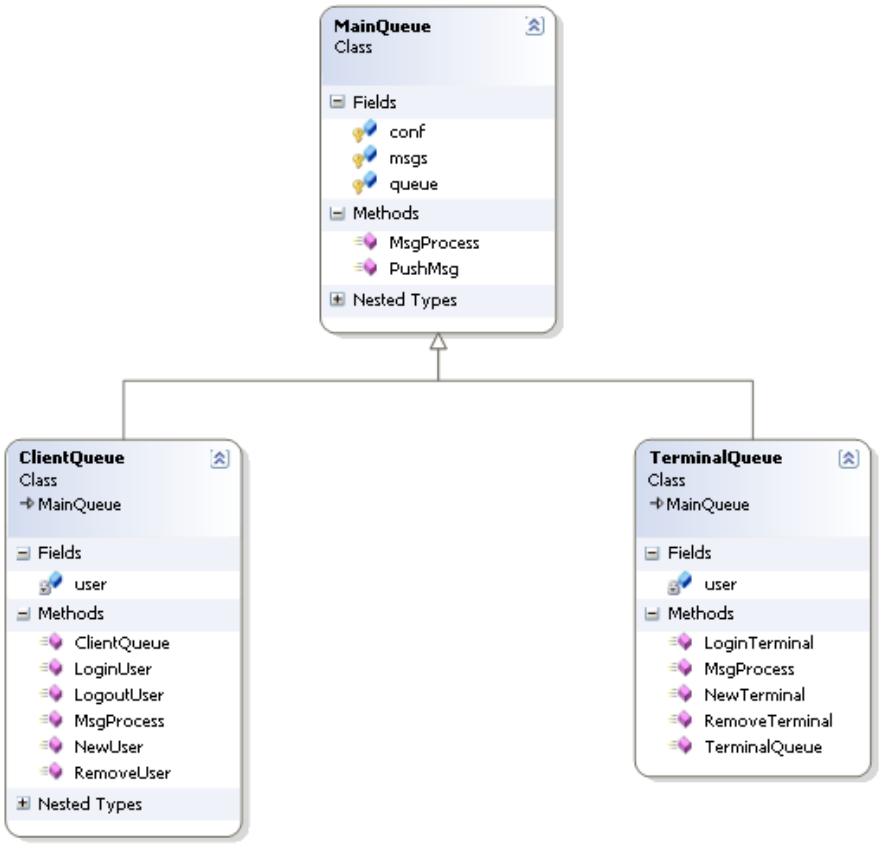
Obr. 6: Diagram tříd - Obsluha síťové komunikace

Protože klientských aplikací může být v systému teoreticky libovolné množství, musí být zajištěna schopnost serveru zpracovávat komunikaci se všemi klienty současně a přitom ještě vykonávat úkony potřebné k řízení procesů. Pro připojení klientů bude tedy na serveru otevřen jeden port, na který se budou připojovat všichni, a pro zaručení konkurentnosti bude komunikace s nimi probíhat asynchronně. Pro každého připojeného klienta bude otevřen nový socket, aby neblokoval připojení ostatních klientů a současně bude umožněno identifikovat jednotlivé klienty právě podle tohoto socketu. Takto jsme schopni rozlišovat jednotlivé klienty od sebe, avšak pro zajištění bezpečnosti budou v tuto chvíli mít možnost odeslat pouze autentizační údaje, veškerá ostatní komunikace bude ignorována. Autentizace proběhne zprávou `LOGIN` (viz. Příloha 3), která obsahuje uživatelské jméno a heslo uživatele pracujícího s daným klientem. Toto je ověřeno proti databázi a pokud kombinace jméno-heslo odpovídá, je klientovi umožněna komunikace podle jeho oprávnění.

Aby byla zajištěna plynulost komunikace i při větším množství klientů a nehrozily ztráty zpráv, či přehlcení serveru zprávami, je na straně příjmu zpráv zavedena fronta zpráv (FIFO), kam se každá příchozí zpráva nejprve uloží a server si je odtud poté sekvenčně odebírá. S takovým řešením je

možné sledovat vytížení, resp. přetížení serveru, které by bylo indikováno stále vzrůstajícím počtem zpráv ve frontě, které by server nebyl schopen dostatečně rychle zpracovat. Pokud by např. bylo ve frontě již příliš velké množství zpráv, server může pozastavit jejich řazení do fronty, každou příchozí zahodit a pouze odeslat nazpět zprávu o přetížení serveru.

V provozu se poté nabízejí dvě možnosti způsobu upozorňování klientů na dostupné kroky a procesy. Buďto se budou klienti periodicky dotazovat serveru nebo server, vždy při připojení klienta nebo vzniku nového kroku či procesu, automaticky bude rozesílat patřičné informace.



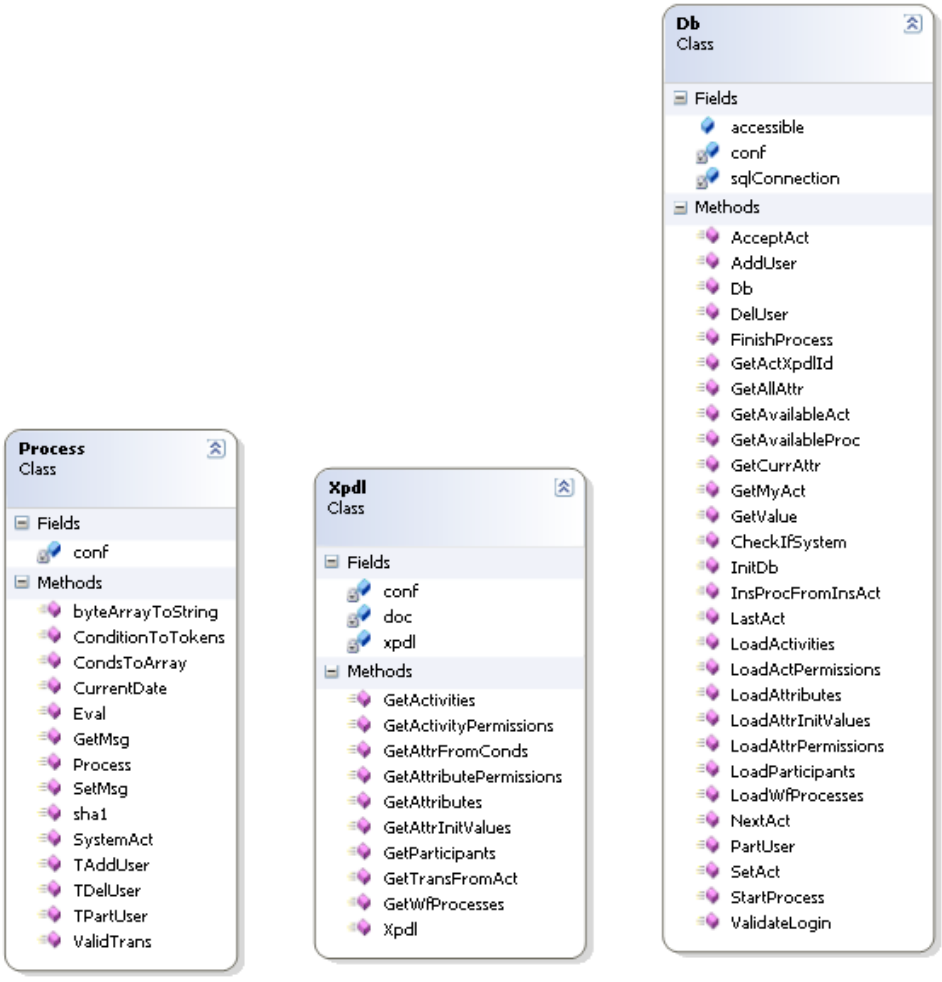
Obr. 7: Diagram tříd - manipulace se zprávami

Z důvodu bezpečnosti a rozšíření možností řízení bude připojení administračního nástroje serveru realizováno přes odlišný port než připojení klientů. Tento port, přestože bude technicky řešen stejně jako spojení s klienty, bude umožňovat pouze jedno spojení najednou. Stejně jako v případě klientů bude vyžadována autentizace, která však nebude ověřována přes databázi, ale heslo pro administrátora bude uloženo v konfiguračním souboru serveru. Takové řešení umožní připojení administrátora na server i v případě, kdy databáze nebude dostupná. Administrátor bude mít

přiřazenou vlastní frontu zpráv, aby nemohl nastat případ, kdy kvůli zahlcení požadavky klientů nebude možné server řídit. Stejně tak to umožňuje snadné nastavení priorit, kdy všechny administrátorské zprávy budou vyřizovány přednostně před klientskými.

3.3 Jádro serveru

Výkonná část serveru má na starost správu procesních definic, databáze, vyřizování zpráv od klientů a administrátora a řízení běhu instancí procesů. V praxi by bylo vhodné implementovat i algoritmy kontroly konzistence databáze, zálohování databáze, aktualizace procesních definic, což ale není součástí této práce. Třídy, ze kterých se jádro skládá jsou znázorněny na Obr. 8.



Obr. 8: Diagram tříd - jaderné třídy pro obsluhu databáze (Db), XPDL dokumentů (Xpdl) a vyřízení zpráv a dalších pomocných činností (Process)

V této práci nebude implementováno řízení deadline procesu, možnost přiložení dokumentu k procesu a možnost nabídnout krok ke zpracování pouze konkrétnímu uživateli.

Třída *Db* zajišťuje obsluhu databáze, uchovává informaci o připojení, dostupnosti databáze a také všechny metody pro práci s ní. Třída *Xpdl* čte podle potřeby data z XPDL dokumentů a předává je systému. *Process* poté obsahuje metody obsluhující jednotlivé zprávy a také podpůrné statické metody.

Jádro serveru využívá tři programová vlákna. První je použito pro inicializaci serveru, tedy načtení konfiguračního souboru, XPDL dokumentů, kontrolu databáze, vytvoření všech objektů pro obsluhu síťových služeb. Další činnost je již řízena událostmi na otevřených socketech, které řeší příjem a odesílání zpráv a přijaté zprávy řadí do odpovídající fronty. Jakmile je libovolná fronta neprázdná, spustí se v novém vlákně metoda zajišťující obsluhu dané fronty a ta postupně vyřídí všechny uložené zprávy.

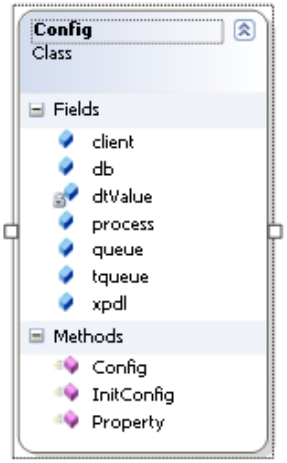
V praxi server musí běžet jako služba, jinak by byl jeho běh závislý na přihlášeném uživateli do systému, což není vhodné. V této implementaci však bude server spouštěn jako normální uživatelský proces.

3.4 Konfigurační soubor

Konfigurační soubor v textové podobě umožňuje definovat základní vlastnosti a nastavení serveru před jeho samotným spuštěním. Každý řádek konfiguračního souboru představuje jednu položku. Formát je následující:

```
# komentář na řádku začínajícím mřížkou
nazev_polozky=hodnota
```

Soubor obsahuje informace o cestě ke XPDL dokumentům, informace o databázi, číslech portů, kde má server naslouchat příchozím spojením atd. Obsluhovaný je třídou *Config*, viz. Obr. 9 Vzorový konfigurační soubor i s vysvětlujícími komentáři je na CD viz. Příloha 5.



Obr. 9: Diagram tříd - Obsluha konfiguračního souboru

4 Implementace systému

4.1 Použitá platforma

Implementace je realizována technologií .NET nejnovější verze 3.5, od společnosti Microsoft Corporation. Tato technologie byla zvolena pro její multiplatformnost a moderní pojetí. Podobně jako třeba Java je totiž zdrojový kód nejprve přeložen do tzv. bytekódu a do samotného strojového kódu se přeloží až při spuštění na konkrétním stroji pomocí *Common Language Runtime*, který zajistí správný překlad pro daný operační systém a CPU[1].

.NET 3.5 podporuje operační systémy Windows 98 a vyšší, včetně nejnovějšího Windows 7[2]. Operační systémy Windows dnes pokrývají drtivou většinu trhu, je tedy zajištěna dobrá dostupnost platformy[3].

Jako databázový systém je použit SQL Server 2005. Produkt je, stejně jako operační systém, od společnosti Microsoft, čímž je teoreticky zajištěna lepší spolehlivost, než v případě jiných databází. Současně lze pro menší řešení využít SQL Server 2005 ve verzi Express, která je distribuována zdarma. Express verze obsahuje několik omezení, jako například omezení velikosti databáze, a nelze ji proto použít vždy.

4.2 Práce s XPDL soubory

Současná implementace serveru načítá XPDL soubory při spuštění do databáze, pokud je zadán parametr `loadxpdl`. Takové spuštění potom vypadá následovně:

```
# WFServer.exe /cesta/ke/konfiguracnimu/souboru loadxpdl
```

Pokud jsou XPDL soubory již načteny, server se spustí bez druhého parametru. Toto řešení je čistě účelové, v praxi bude třeba, aby server sám poznal, co má načítat a co už načteno má. Toto by

1 *Msdn : Visual Studio 2008 Developer Center* [online]. Microsoft Corporation, c2009 [cit. 2009-05-03]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/default.aspx>.

2 Tamtéž

3 MICK, Jason. Mac OS X Hits Record Marketshare, Continues to Chip at Windows Lead. DailyTech [online]. 2008 [cit. 2009-05-15]. Dostupný z WWW: <http://www.dailytech.com/Mac+OS+X+Hits+Record+Marketshare+Continues+to+Chip+at+Windows+Lead/article10559.htm>.

šlo řešit např. uložením kontrolního součtu z některé hashovací funkce, čímž by server hned viděl, které soubory se změnili a že má jejich obsah porovnat s databází.

Relevantní informace o procesu jsou z nich načteny do databáze. Vynechány jsou pouze přechody mezi kroky – `Transitions`. Ty se čtou ze souboru až v případě potřeby.

Protože XPDL umožňuje elementem `ExternalPackages` definovat umístění dalších XPDL souborů, které k tomuto patří, server nejprve otevře soubor, který má definovaný položkou `xpdl_filename` v konfiguračním souboru a v něm zkontroluje přítomnost elementu `ExternalPackages`. Pokud jej najde, načte z něj názvy souborů, jejichž umístění je definováno opět v konfiguračním souboru a to položkou `xpdl_dir`. Všechny tyto soubory si načte do paměti jako jeden dokument.

Pokud existuje neukončená instance procesu, je změna definice procesu možná pouze omezeně, a to na úrovni cest mezi kroky, které se do databáze nenačítají. Po jejich úpravě v definičním souboru je třeba zkontrolovat, zda žádná z instancí daného procesu není v kroku, ze kterého by se po provedené změně již nemohla dostat a také zda jsou nastavena všechna potřebná oprávnění k atributům, protože tato se automaticky nastavují pouze při prvním načtení definic. Nakonec je třeba server restartovat, aby se upravená definice dostala do programové paměti.

4.3 Komunikace s administračním rozhranním

Pro komunikaci je využito síťového protokolu TCP/IP a to z důvodu možnosti vzdáleného připojení k serveru a současně k fyzickému oddělení jádra serveru od administračního rozhranní, čímž toto může být modifikováno naprosto nezávisle, stejně jako může vzniknout více typů administračních rozhranní, třeba podle způsobu použití. V praxi by se z hlediska bezpečnosti uplatnilo použití šifrované komunikace protokolem SSL.

Vzhledem k asynchronnímu přenosu dat, je stanoven terminální znak zprávy `'$'`, aby bylo pro přijímací stranu zřejmé, že zprávu již přijala celou. Zpráva přijatá serverem se zařadí do fronty ke zpracování až po příjmu tohoto terminálního znaku. Současně se může připojit jedno administrační rozhranní. Toto spojení je trvalé, není tedy při nečinnosti po uplynutí časového limitu odpojeno, což by bylo v praxi užitečné, pokud by se např. administrátor někde zapomněl odhlásit a poté se chtěl připojit z jiného místa.

Samotná komunikace je řešena textovým protokolem. Implementované zprávy jsou tyto:

`LOGIN`	autentizace administrátora
`ADDUSER`	přidání uživatele do systému
`PARTUSER`	přiřazení uživatele do rolí

Specifikace jednotlivých zpráv viz. Příloha 3.

4.4 Komunikace s klientskou aplikací

Komunikace s klienty je řešena obdobně jako je tomu v případě administračního nástroje, ale na rozdíl od něj není omezen počet připojení a současně tedy může být připojeno libovolné množství klientů. Jejich počet je omezen pouze nastavením operačního systému, kde server běží.

Implementované zprávy jsou:

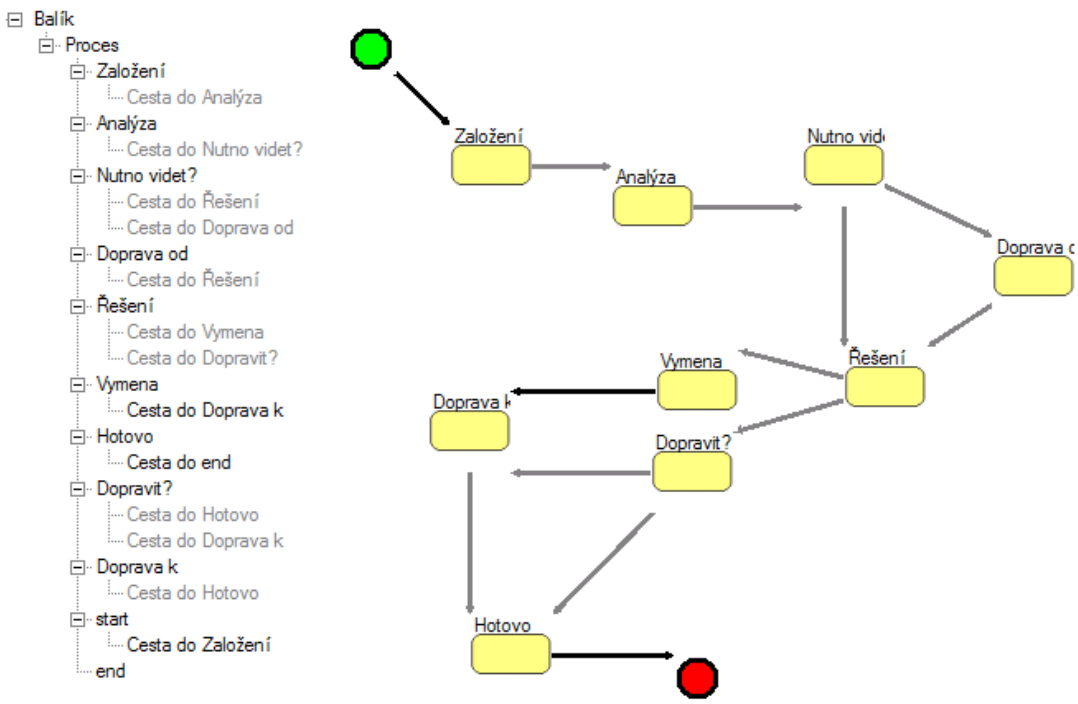
`LOGIN`	autentizace uživatele
`LOGOUT`	odhlášení uživatele ze systému (socket zůstává připojený)
`GET AVAILABLE PROC`	vyžádání seznamu procesů, které uživatel může založit
`GET AVAILABLE ACT`	vyžádání seznamu kroků, které uživatel může přijmout
`GET ALL ATTR id`	vyžádání seznamu atributů k instanci kroku
`GET CURR ATTR idp ida`	vyžádání seznamu atributů ke konkrétní instanci kroku
`GET MY ACT [idp]`	vyžádání seznamu kroků přiřazených uživateli
`START`	založení workflow
`ACCEPT`	příjem kroku ke zpracování

4.5 Příklad vykonání procesu

XPDL kód tohoto procesu je přiložen v příloze 3 a také v elektronické podobě v příloze 4.

Proces představuje řešení technického problému zákazníka s hardwarovým produktem v servisním středisku. Diagram tohoto procesu znázorňuje Obr. 10, tak jak zobrazí modelovací nástroj z bakalářské práce Zdeňka Soukupa[1].

1 SOUKUP, Zdeněk: *Systém pro podporu řízení workflow*, bakalářská práce, Brno, FIT VUT v Brně, 2009



Obr. 10: Vzorový proces - Vyřízení reklamace

Jelikož jde o demonstrační příklad, uvedu zde i úkony, které musí provést administrátor, než je možné proces použít.

4.5.1 Příprava procesu

Protože XPDL definice nemůže obsahovat informace o uživatelích systému, o jejich pracovním zařazení a oprávněních, musí být toto nejprve zadáno do systému administrátorem. Vzorový proces nám definuje 4 role, které proces zpracovávají. Jsou to: operátor, technik, dopravce a automat. Automat je speciální role typu systém, kterou reprezentuje samotný server, a je přiřazena pouze prázdným krokům, které slouží ke zpřehlednění diagramu a zjednodušení zápisu přechodových podmínek. Této roli se tedy nepřiřazují žádní uživatelé.

U ostatních rolí to však je potřeba udělat. Administrátor tedy odešle na server tuto sekvenci zpráv:

```
ADDUSER pavel aaa Pavel Plekanec$
PARTUSER pavel operator$
ADDUSER martin aaa Martin Krstev$
PARTUSER martin technik$
```

```
ADDUSER karel aaa Karel Korbanov$
PARTUSER karel dopravce$
```

Tímto byli vytvořeni uživatelé Pavel, Martin a Karel, z nichž každý zastává jednu roli. Všichni mají pro jednoduchost nastaveno stejné heslo do systému: aaa. V tuto chvíli je proces načten do databáze (vykonáno při spuštění serveru, viz. kapitola 4.2) a všechny role mají přiřazené uživatele. Pokud tedy není proces v databázi označen jako neaktivní, nic uživatelům nebrání v jeho používání.

4.5.2 Zpracování uživateli

Jakmile zákazník zavolá na linku technické podpory, operátor Pavel vyslechne jeho potíž, přihlásí se do systému a založí workflow.

Zpráva uživatele	Odpověď serveru
`LOGIN pavel aaa$`	
	`LOGIN DONE$`
`GET AVAILABLE PROC$`	
	`AVAILABLE PROC LIST 1;pkg1_wfp1;NULL;$`
`START 1$`	
	`START DONE 1$`

Současně Pavel přijme první krok – Založení, a vyplní potřebné atributy podle informací od zákazníka.

`GET AVAILABLE ACT$`	
	`AVAILABLE ACT LIST 1;Zalozeni;NULL;1;$`
`ACCEPT 1$`	
	`ACCEPT DONE$`
`GET CURR ATTR 1 1$`	
	`CURR ATTR LIST 1;"zakaznik";"udaje o zakaznikovi.";rw;STRING;NULL;1;2;"popis od zakaznika";"Popis problemu dodany zakaznikem.";rw;STRING;NULL;1;4;"produkt";"O jaky produkt se jedna?";rw;STRING;NULL;1;$`
`SET ACT 1 ATTR LIST 1;"Patrik Bicanek, IQ COMP, a.s.";2;"Obraz je stale cerny,`	

```
sviti akorat dioda indikujici zapnuty
stav";4;"LCD monitor";$
                                    ACT DONE$
```

Tímto práce Pavla končí a proces se přesunuje k technikovi Martinovi, jehož úkolem je analyzovat problém a rozhodnout další postup řešení.

```
LOGIN martin aaa$
                                    LOGIN DONE$
GET AVAILABLE ACT$
                                    AVAILABLE ACT LIST 2;Analyza;NULL;1;$
ACCEPT 2$
                                    ACCEPT DONE$
GET CURR ATTR 1 2$
                                    CURR ATTR LIST 5;"nutno videt";"Je
                                    nutne produkt
                                    dodat?";rw;STRING;NULL;1;$
SET ACT 2 ATTR LIST 5;"ANO";$
                                    ACT DONE$
```

Jelikož reklamovaným produktem je LCD monitor, nemůže opravu provést na dálku. Nechá si tedy monitor přivést do servisu.

```
GET CURR ATTR 1 2$
                                    CURR ATTR LIST 5;"nutno videt";"Je
                                    nutne produkt
                                    dodat?";rw;STRING;NULL;1;$
SET ACT 2 ATTR LIST 5;"ANO";$
                                    ACT DONE$
```

Tímto se tok procesu dostává k dopravci Karlovi, jehož úkolem je produkt přivést do servisního střediska.

```
LOGIN karel aaa$
                                    LOGIN DONE$
GET AVAILABLE ACT$
                                    AVAILABLE ACT LIST 4;Doprava
                                    od;NULL;1;$
```

```
ACCEPT 4$
                                    ACCEPT DONE$
GET CURR ATTR 1 4$
                                    CURR ATTR LIST 9;"doruceno od";"Byl
                                    produkt privezen od zakaznika v
                                    poradku?";rw;STRING;NULL;1;6;"adresa
                                    zakaznika";"Adresa, odkud bude dodan
                                    produkt.";rw;STRING;NULL;1;$
SET ACT 4 ATTR LIST 9;"ANO";
6;"Andrysova 16, Tisnov";$
                                    ACT DONE$
```

V tuto chvíli je již monitor v servisu, tok procesu se tak opět přesouvá k technikovi Martinovi. Ten provede analýzu produktu, zjistí, že jde o vadnou katodu v podsvícení a provede její výměnu.

```
GET AVAILABLE ACT$
                                    AVAILABLE ACT LIST 5;Vyreseno?;NULL;1;$
ACCEPT 5$
                                    ACCEPT DONE$
GET CURR ATTR 1 5$
                                    CURR ATTR LIST 7;"vyreseno";"Byl
                                    problem vyresen?";rw;STRING;NULL;1;
                                    8;"popis reseni";"Popis, jak byl
                                    problem vyresen.";rw;STRING;NULL;1;$
SET ACT 5 ATTR LIST 7;"ANO";
8;"Vadna katoda, provedena jeji
vymena.";$
                                    ACT DONE$
```

Produkt je tak připraven na cestu zpět k zákazníkovi. Čehož se ujímá Karel.

```
GET AVAILABLE ACT$
                                    AVAILABLE ACT LIST 7;Doprava k;NULL;1;$
ACCEPT 7$
                                    ACCEPT DONE$
GET CURR ATTR 1 7$
                                    CURR ATTR LIST 10;"doruceno k";"Byl
                                    produkt odvezen k zakaznikovi v
                                    poradku?";rw;STRING;NULL;1;$
```

```
SET ACT 7 ATTR LIST 10;"ANO";$
                                        ACT DONE$
```

Po dokončení posledního kroku je workflow automaticky označeno jako ukončené.

5 Závěr

Cílem této práce byl návrh a implementace serverové části systému pro podporu workflow a její spolupráce s designérem pro modelování procesů z bakalářské práce Zdeňka Soukupa[1].

Současná implementace je modelová, velice dobře však posloužila k ověření správnosti navrženého konceptu. Systém je schopen řídit jednoduché procesy, na kterých byla testována spolehlivost a odhalována slabá místa. Teoreticky se může připojit libovolné množství klientů a na serveru může běžet libovolné množství workflow. Toto je omezeno pouze výkonem stanice, kde serverová služba běží, výkonem databáze, síťovou propustností a případnými omezeními operačního systému. K nim patří počet TCP spojení, velikost operační paměti a velikost operační paměti, která může být přidělena jednomu procesu. Současná síťová komunikace přes textové protokoly je omezena na ASCII znaky, není tedy možné posílat znaky s českou diakritikou.

Databáze udržuje veškerá data a proto odstavení serveru a jeho opětovné nastartování nezpůsobí vznik žádných nekonzistentních stavů. Toto restartování dokonce umožňuje drobné změny v definici procesu. Týká se to však jen cest (`Transitions`), které nejsou uchovávány v databázi. Při tomto je ale třeba postupovat obezřetně vzhledem ke stavu běžících workflow, aby nedošlo k jejich zablokování. Také je třeba hlídat nastavení oprávnění ke krokům, protože ty se ukládají do databáze pouze při prvním načítání definice.

Velice dobře se osvědčil také systém s frontami příchozích zpráv, který funguje spolehlivě a pokud by se implementovali i prvky ochrany proti přetížení (zahazováním zpráv a informováním klientů o přetížení), byl by dobře použitelný. Fronta také určuje pořadí zpracování zpráv, protože je současně použito pouze jedno databázové spojení. Protokoly, v jejichž formátu jsou zprávy posílány, specifikují a implementují omezené množství zpráv nezbytné pro vykonání procesů. Pro plnohodnotný chod systému a možnost navrhnout dostatečně přívětivou klientskou aplikaci je třeba některé zprávy doimplementovat.

Server je schopen zpracovat XPDL dokumenty vystupující z modelovacího nástroje Zdeňka Soukupa[2], který je součástí systému. Specifikace XPDL je použita ve verzi 1.0 z roku 2002 z důvodu její velké rozšířenosti. Nejnovější verze 2.1 je s touto zpětně kompatibilní, obsahuje více možností pro modelování, a nabízí se tak její implementace v další fázi vývoje.

---

1 SOUKUP, Zdeněk: *Systém pro podporu řízení workflow*, bakalářská práce, Brno, FIT VUT v Brně, 2009

2 Tamtéž.

Literatura

CARDA, Antonín, KUNSTOVÁ, Renáta. *Workflow : Nástroj manažera pro řízení podnikových procesů.* 2. rozš. vyd. Praha : Grada, 2003. 156 s. Management v informační společnosti. ISBN 80-247-0666-0.

DUBRAY, Jean-Jacques. *Critical comments on XPDL 1.0* [online]. c2007 [cit. 2009-04-24]. Dostupný z WWW: <http://www.ebpml.org/xpdl.htm>.

HAVEY, Michael. *Essential business process modeling*. [s.l.] : O\'Reilly, 2005. 332 s. ISBN 9780596008437.

MARGUERIE, Fabrice, EICHERT, Steve, WOOLEY, Jim. *LINQ in Action*. [s.l.] : Wiley-India, 2008. 602 s. ISBN 9788177228823.

MICK, Jason. Mac OS X Hits Record Marketshare, Continues to Chip at Windows Lead. DailyTech [online]. 2008 [cit. 2009-05-15]. Dostupný z WWW: <http://www.dailytech.com/Mac+OS+X+Hits+Record+Marketshare+Continues+to+Chip+at+Windows+Lead/article10559.htm>.

Msdn : Visual Studio 2008 Developer Center [online]. Microsoft Corporation, c2009 [cit. 2009-05-03]. Dostupný z WWW: <http://msdn.microsoft.com/en-us/library/default.aspx>.

ROB, Peter, CROCKETT, Keeley, CORONEL, Carlos. *Database Systems.* [s.l.] : Cengage Learning EMEA, 2008. 832 s. ISBN 9781844807321.

ROZENBERG, Grzegorz. Advanced Course on Petri Nets. [s.l.] : Springer, 2004. 849 s. ISBN 9783540222613.

SOUKUP, Zdeněk: *Systém pro podporu řízení workflow*, bakalářská práce, Brno, FIT VUT v Brně, 2009.

WHITE, Steven A. *Introduction to BPMN*. IBM Corporation [online]. 2004 [cit. 2009-04-23]. Dostupný z WWW: <http://www.bpmn.org/Documents/Introduction to BPMN.pdf>.

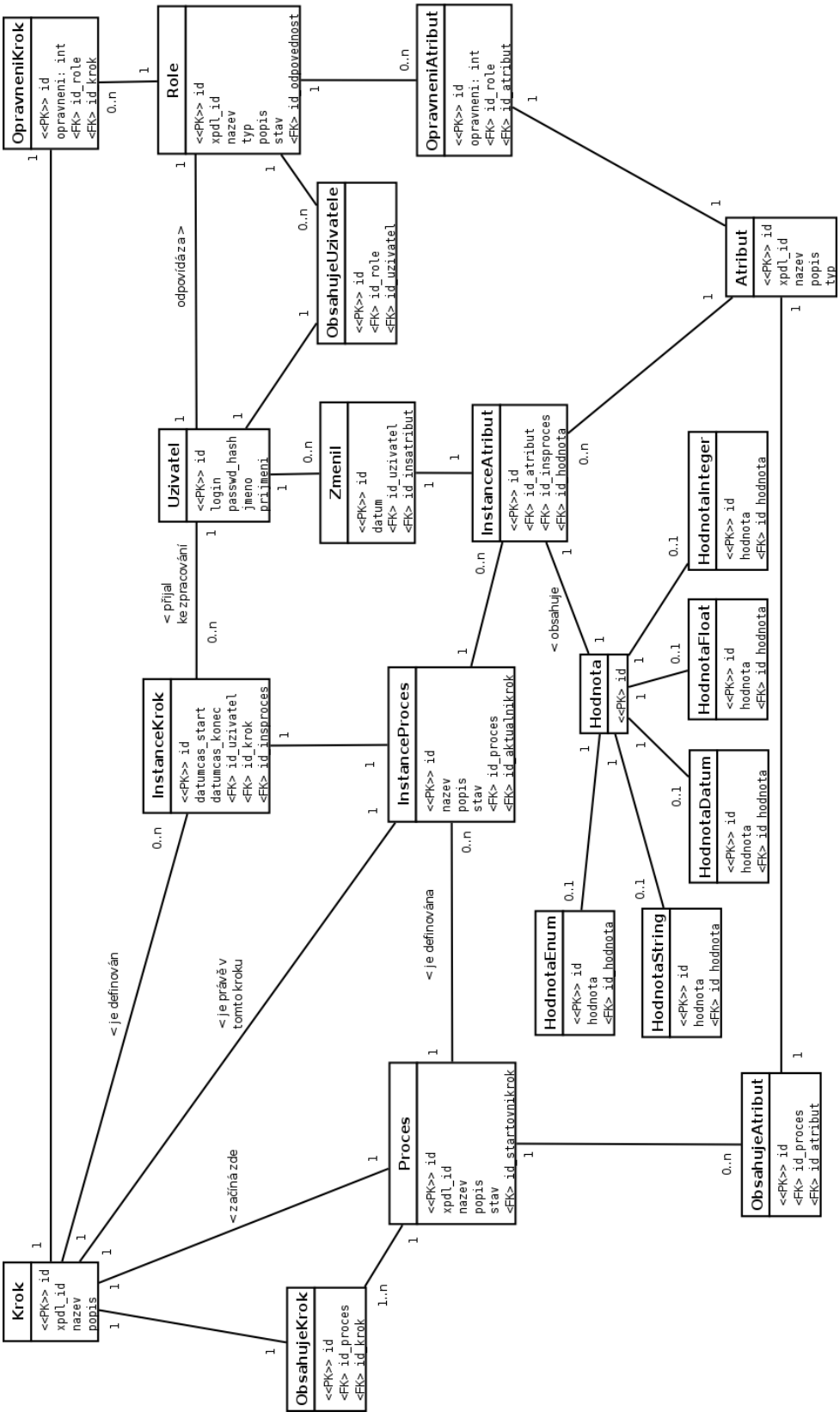
Workflow Management Coalition Workflow Standard : *Workflow Process Definition Interface -- XML Process Definition Language* [online]. Workflow Management Coalition, 2002, October 25, 2002 [cit. 2009-04-24].
Dostupný z WWW: <http://xml.coverpages.org/XPDLv10.pdf>.

XPDL Support and Resources [online]. Workflow Management Coalition, c2009 [cit. 2009-05-10]. Dostupný z WWW: <http://www.wfmc.org/xpdl.html>.

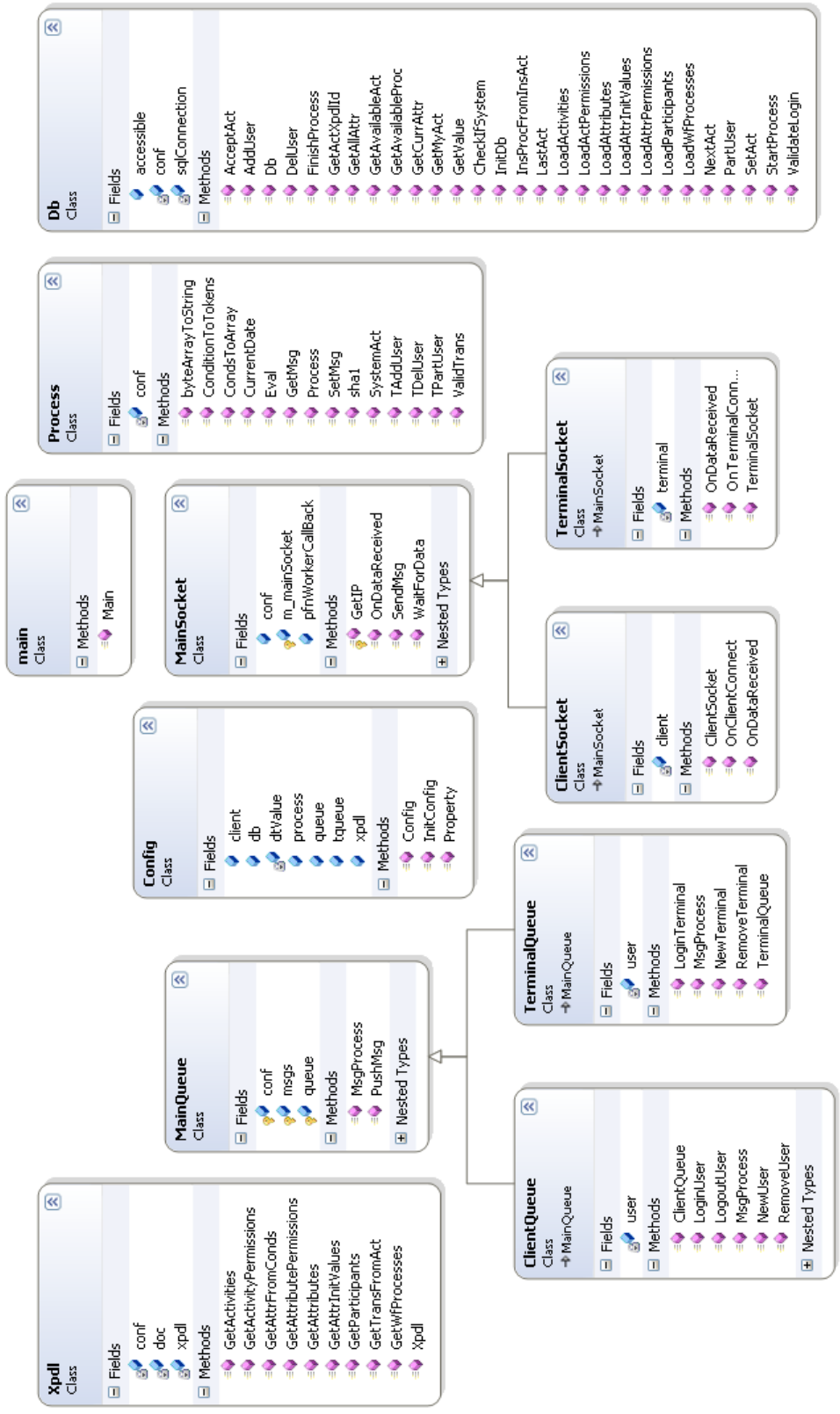
Seznam ilustrací

Seznam příloh

Příloha 1: ER diagram



Příloha 2: Diagram tříd



Příloha 3: Specifikace komunikačních protokolů

Terminální znak všech zpráv je '$'.

Řetězcové konstanty se uzavírají do uvozovek – "řetězec".

Komunikace s administračním rozhranním

Přihlášení terminálu

```
LOGIN passwd
```

passwd	heslo

Odpovědi serveru:

LOGIN DONE	přihlášení proběhlo úspěšně
LOGIN FAILED	přihlášení bylo neúspěšné, zadáno chybné heslo
LOGIN SYNTAX	chybná syntaxe zprávy

Přidání uživatele do systému

```
ADDUSER login passwd jméno příjmení [příjmení_2]
```

login	přihlašovací jméno uživatele
heslo	heslo uživatele, SHA1 hash uložena v databázi
jméno	křestní jméno
příjmení [příjmení_2]	příjmení uživatele, volitelně lze zadat i druhé příjmení

Odpovědi serveru:

ADDUSER DONE	přidání uživatele proběhlo v pořádku
ADDUSER LOGIN EXISTS	uživatelské jméno již v databázi existuje
ADDUSER SYNTAX	chybná syntaxe zprávy
ADDUSER ERROR	přidání uživatele nelze provést (nedostupná db)

Přiřazení rolí uživateli

```
PARTUSER login participant [participant] …
```

`login`	přihlašovací jméno uživatele
`participant`	název role, do které má být uživatel přiřazen. Rolí lze zadat libovolný počet, minimum je však jedna.

Odpovědi serveru:

`PARTUSER DONE`	přiřazení rolí proběhlo v pořádku
`PARTUSER INVALID LOGIN`	zadané uživatelské jméno neexistuje
`PARTUSER INVALID [participant]`	role `[participant]` v systému neexistuje
`PARTUSER SYNTAX`	chybná syntaxe zprávy
`PARTUSER ERROR`	přiřazení rolí nelze provést (nedostupná db)

Komunikace s klientskou aplikací

Přihlášení uživatele

```
LOGIN login passwd
```

`login`	uživatelské jméno
`passwd`	heslo

Odpovědi serveru:

`LOGIN DONE`	přihlášení proběhlo úspěšně
`LOGIN FAILED`	přihlášení bylo neúspěšné, zadáno chybné heslo
`LOGIN SYNTAX`	chybná syntaxe zprávy
`LOGIN ERROR`	přihlášení uživatele nelze provést (nedostupná db)

V případě, že neautentizovaný uživatel odešle zprávu jinou než `LOGIN`, bude tato ignorována a v odpověď mu přijde zpráva `LOGIN FIRST`.

Odhlášení uživatele

```
LOGOUT
```

Odpověď serveru:

`LOGOUT DONE` uživatel úspěšně odhlášen

Vyžádání seznamu spustitelných procesů

`GET AVAILABLE PROC`

Odpovědi serveru:

`AVAILABLE PROC LIST id;název;[popis | NULL]; …` seznam dostupných procesů

- `id` identifikace procesu v rámci databáze
- `název` název procesu
- `popis` stručný popis procesu, volitelný parametr

`AVAILABLE PROC NULL` uživatel nemá oprávnění oprávnění k založení žádného workflow

`GET ERROR` nelze provést (nedostupná db)

Vyžádání seznamu kroků k přijmutí

`GET AVAILABLE ACT`

Odpovědi serveru:

`AVAILABLE ACT LIST id;název;[popis | NULL];id_insprocesu; …`

- `id` identifikace kroku v rámci databáze
- `název` název kroku
- `popis` stručný popis kroku, volitelný parametr
- `id_insprocesu` identifikátor instance procesu, kterému krok náleží

`AVAILABLE ACT NULL` žádný krok k přijmutí není k dispozici

`GET ERROR` nelze provést (nedostupná db)

Vyžádání seznamu všech atributů v procesu

`GET ALL ATTR id_insproces`

- `id_insproces` identifikátor instance procesu, jehož atributy žádáme

Odpovědi serveru:

`ALL ATTR LIST id;nazev;[popis | NULL];[r | rw];typ;[hodnota | NULL];id_insproces; …`

- `id` identifikátor instance atributu

`nazev`	nazev atributu
`popis`	popis atributu (může být prázdný – `NULL`)
`r | rw`	oprávnění k atributu, `r` – čtení, `rw` – čtení/zápis
`typ`	datový typ atributu
`hodnota`	současná hodnota atributu, pokud není nastavena – `NULL`
`id_insproces`	identifikátor instance procesu
`ALL ATTR NULL`	žádný atribut není k dispozici (instance procesu neexistuje nebo uživatel nemá k žádnému atributu oprávnění)
`GET ERROR`	nelze provést (nedostupná db)

Vyžádání seznamu atributů k přijatému kroku

`GET CURR ATTR id_insproces id_inskrok`

`id_insproces`	identifikátor instance procesu, jehož atributy žádáme
`id_inskrok`	identifikátor instance kroku, jehož atributy žádáme (krok musí být přiřazen uživateli)

Odpovědi serveru:

`CURR ATTR LIST id;nazev;[popis | NULL];[r | rw];typ;[hodnota | NULL];id_insproces; …`

`id`	identifikátor instance atributu
`nazev`	nazev atributu
`popis`	popis atributu (může být prázdný – `NULL`)
`r | rw`	oprávnění k atributu, `r` – čtení, `rw` – čtení/zápis
`typ`	datový typ atributu
`hodnota`	současná hodnota atributu, pokud není nastavena – `NULL`
`id_insproces`	identifikátor instance procesu
`CURR ATTR NULL`	žádný atribut není k dispozici (instance procesu nebo kroku neexistuje nebo nemá krok přiřazen)
`GET ERROR`	nelze provést (nedostupná db)

Vyžádání seznamu neukončených kroků přijatých uživatelem

`GET MY ACT [id_insproces]`

`id_insproces`	identifikátor instance procesu, volitelný parametr

Odpovědi serveru:

`MY ACT LIST id_inskrok;id_insproces; …`

`id_inskrok`	identifikátor instance kroku
`id_insproces`	identifikátor instance procesu

`MY ACT NULL`	žádný krok není uživateli přiřazen
`GET ERROR`	nelze provést (nedostupná db)

Založení workflow

`START id [nazev] [popis]`

`id`	identifikátor procesu v databázi (vrací `GET AVAILABLE PROC`)
`nazev`	volitelný parametr specifikující název workflow
`popis`	volitelný parametr specifikující popis workflow

Odpovědi serveru:

`START DONE id`	workflow založeno s identifikátorem `id`
`START PERMISSIONS`	nedostatečná oprávnění pro založení workflow
`START SYNTAX`	chybná syntaxe zprávy
`START ERROR`	založení workflow nelze provést (nedostupná db)

Příjem kroku ke zpracování

`ACCEPT id`

`id`	identifikátor instance kroku (vrací `GET AVAILABLE ACT`)

Odpovědi serveru:

`ACCEPT DONE`	krok úspěšně přiřazen
`ACCEPT PERMISSIONS`	nedostatečná oprávnění k přijmutí kroku
`ACCEPT TAKEN`	krok je již zpracováván jiným uživatelem
`ACCEPT SYNTAX`	chybná syntaxe zprávy
`ACCEPT ERROR`	příjem kroku nelze provést (nedostupná db)

Odeslání zpracovaného kroku

`SET ACT id_act ATTR LIST id_attr;hodnota; …`

`id_act`	identifikátor instance kroku

`id_attr`	identifikátor instance atributu
`hodnota`	nová hodnota atributu

Odpovědi serveru:

`ACT DONE`	krok přijat
`ACT WRONG`	instance kroku není přiřazena tomuto uživateli
`ACT NONE`	instance krok neexistuje
`ACT NO ATTR id`	instance atributu neexistuje nebo uživatel nemá oprávnění
`ACT VALUES`	odeslané hodnoty atributů neumožňují přejít do dalšího kroku
`SET ERROR`	nelze provést (nedostupná db)

Příloha 4: Zdrojový kód XPDL dokumentu

```
<?xml version="1.0" encoding="utf-8"?>
<xpdl:Package Id="pkg1" Name="pkg1" xmlns="http://www.wfmc.org/2002/XPDL1.0"
xmlns:xpdl="http://www.wfmc.org/2002/XPDL1.0">
  <xpdl:PackageHeader>
    <xpdl:XPDLVersion>1.0</xpdl:XPDLVersion>
    <xpdl:Vendor>xsouku02</xpdl:Vendor>
    <xpdl:Created>28.4.2009 19:47</xpdl:Created>
  </xpdl:PackageHeader>
  <xpdl:WorkflowProcesses>
    <xpdl:WorkflowProcess Id="pkg1_wfp1" Name="pkg1_wfp1">
      <xpdl:Created>28.4.2009 19:47</xpdl:Created>
      <xpdl:DataFields>
        <xpdl:DataField Id="0" Name="zakaznik">
          <xpdl:DataType>
            <xpdl:BasicType Type="STRING" />
          </xpdl:DataType>
          <xpdl:InitialValue></xpdl:InitialValue>
          <xpdl:Description>Údaje o zákazníkovi.</xpdl:Description>
        </xpdl:DataField>
        <xpdl:DataField Id="1" Name="popis od zakaznika">
          <xpdl:DataType>
            <xpdl:BasicType Type="STRING" />
          </xpdl:DataType>
          <xpdl:InitialValue></xpdl:InitialValue>
          <xpdl:Description>Popis problému dodaný zákazníkem.</xpdl:Description>
        </xpdl:DataField>
        <xpdl:DataField Id="2" Name="popis od technika">
          <xpdl:DataType>
            <xpdl:BasicType Type="STRING" />
          </xpdl:DataType>
          <xpdl:InitialValue></xpdl:InitialValue>
          <xpdl:Description>Technický popis problému.</xpdl:Description>
        </xpdl:DataField>
        <xpdl:DataField Id="3" Name="produkt">
          <xpdl:DataType>
            <xpdl:BasicType Type="STRING" />
          </xpdl:DataType>
          <xpdl:InitialValue></xpdl:InitialValue>
          <xpdl:Description>O jaký produkt se jedná?</xpdl:Description>
        </xpdl:DataField>
        <xpdl:DataField Id="4" Name="nutno videt">
          <xpdl:DataType>
            <xpdl:BasicType Type="STRING" />
          </xpdl:DataType>
          <xpdl:InitialValue></xpdl:InitialValue>
          <xpdl:Description>Je nutné produkt dodat?</xpdl:Description>
        </xpdl:DataField>
        <xpdl:DataField Id="5" Name="adresa zakaznika">
          <xpdl:DataType>
            <xpdl:BasicType Type="STRING" />
          </xpdl:DataType>
          <xpdl:InitialValue></xpdl:InitialValue>
          <xpdl:Description>Adresa, odkud bude dodán produkt.</xpdl:Description>
        </xpdl:DataField>
        <xpdl:DataField Id="6" Name="vyreseno">
          <xpdl:DataType>
            <xpdl:BasicType Type="STRING" />
          </xpdl:DataType>
          <xpdl:InitialValue></xpdl:InitialValue>
          <xpdl:Description>Byl problém vyřešen?</xpdl:Description>
        </xpdl:DataField>
```

```
        <xpdl:DataField Id="7" Name="popis reseni">
          <xpdl:DataType>
            <xpdl:BasicType Type="STRING" />
          </xpdl:DataType>
          <xpdl:InitialValue></xpdl:InitialValue>
          <xpdl:Description>Popis, jak byl problém vyřešen.</xpdl:Description>
        </xpdl:DataField>
        <xpdl:DataField Id="8" Name="doruceno od">
          <xpdl:DataType>
            <xpdl:BasicType Type="STRING" />
          </xpdl:DataType>
          <xpdl:InitialValue></xpdl:InitialValue>
          <xpdl:Description>Byl produkt přivezen od zákazníka v pořádku?
</xpdl:Description>
        </xpdl:DataField>
        <xpdl:DataField Id="9" Name="doruceno k">
          <xpdl:DataType>
            <xpdl:BasicType Type="STRING" />
          </xpdl:DataType>
          <xpdl:InitialValue></xpdl:InitialValue>
          <xpdl:Description>Byl produkt odvezen k zákazníkovi v pořádku?
</xpdl:Description>
        </xpdl:DataField>
      </xpdl:DataFields>
      <xpdl:ActivitySets></xpdl:ActivitySets>
      <xpdl:Activities>
        <xpdl:Activity Id="0" Name="Založení">
          <xpdl:Description></xpdl:Description>
          <xpdl:StartMode>Manual</xpdl:StartMode>
          <xpdl:FinishMode>Manual</xpdl:FinishMode>
          <xpdl:Performer>operator</xpdl:Performer>
          <xpdl:ExtendedAttributes>
            <xpdl:ExtendedAttribute Name="Position">
              <xpdl:xPosition>68</xpdl:xPosition>
              <xpdl:yPosition>75</xpdl:yPosition>
            </xpdl:ExtendedAttribute>
          </xpdl:ExtendedAttributes>
        </xpdl:Activity>
        <xpdl:Activity Id="1" Name="Analýza">
          <xpdl:Description></xpdl:Description>
          <xpdl:StartMode>Manual</xpdl:StartMode>
          <xpdl:FinishMode>Manual</xpdl:FinishMode>
          <xpdl:Performer>technik</xpdl:Performer>
          <xpdl:ExtendedAttributes>
            <xpdl:ExtendedAttribute Name="Position">
              <xpdl:xPosition>166</xpdl:xPosition>
              <xpdl:yPosition>100</xpdl:yPosition>
            </xpdl:ExtendedAttribute>
          </xpdl:ExtendedAttributes>
        </xpdl:Activity>
        <xpdl:Activity Id="2" Name="Nutno videt?">
          <xpdl:Description>Nutno</xpdl:Description>
          <xpdl:StartMode>Automatic</xpdl:StartMode>
          <xpdl:FinishMode>Automatic</xpdl:FinishMode>
          <xpdl:Performer>automat</xpdl:Performer>
          <xpdl:ExtendedAttributes>
            <xpdl:ExtendedAttribute Name="Position">
              <xpdl:xPosition>282</xpdl:xPosition>
              <xpdl:yPosition>75</xpdl:yPosition>
            </xpdl:ExtendedAttribute>
          </xpdl:ExtendedAttributes>
        </xpdl:Activity>
        <xpdl:Activity Id="3" Name="Doprava od">
          <xpdl:Description></xpdl:Description>
          <xpdl:StartMode>Manual</xpdl:StartMode>
```

```
  <xpdl:FinishMode>Manual</xpdl:FinishMode>
  <xpdl:Performer>dopravce</xpdl:Performer>
  <xpdl:ExtendedAttributes>
    <xpdl:ExtendedAttribute Name="Position">
      <xpdl:xPosition>397</xpdl:xPosition>
      <xpdl:yPosition>143</xpdl:yPosition>
    </xpdl:ExtendedAttribute>
  </xpdl:ExtendedAttributes>
</xpdl:Activity>
<xpdl:Activity Id="4" Name="Řešení">
  <xpdl:Description></xpdl:Description>
  <xpdl:StartMode>Manual</xpdl:StartMode>
  <xpdl:FinishMode>Manual</xpdl:FinishMode>
  <xpdl:Performer>technik</xpdl:Performer>
  <xpdl:ExtendedAttributes>
    <xpdl:ExtendedAttribute Name="Position">
      <xpdl:xPosition>307</xpdl:xPosition>
      <xpdl:yPosition>211</xpdl:yPosition>
    </xpdl:ExtendedAttribute>
  </xpdl:ExtendedAttributes>
</xpdl:Activity>
<xpdl:Activity Id="5" Name="Vymena">
  <xpdl:Description></xpdl:Description>
  <xpdl:StartMode>Automatic</xpdl:StartMode>
  <xpdl:FinishMode>Automatic</xpdl:FinishMode>
  <xpdl:Performer>automat</xpdl:Performer>
  <xpdl:ExtendedAttributes>
    <xpdl:ExtendedAttribute Name="Position">
      <xpdl:xPosition>193</xpdl:xPosition>
      <xpdl:yPosition>213</xpdl:yPosition>
    </xpdl:ExtendedAttribute>
  </xpdl:ExtendedAttributes>
</xpdl:Activity>
<xpdl:Activity Id="6" Name="Hotovo">
  <xpdl:Description></xpdl:Description>
  <xpdl:StartMode>Manual</xpdl:StartMode>
  <xpdl:FinishMode>Manual</xpdl:FinishMode>
  <xpdl:Performer>automat</xpdl:Performer>
  <xpdl:ExtendedAttributes>
    <xpdl:ExtendedAttribute Name="Position">
      <xpdl:xPosition>80</xpdl:xPosition>
      <xpdl:yPosition>375</xpdl:yPosition>
    </xpdl:ExtendedAttribute>
  </xpdl:ExtendedAttributes>
</xpdl:Activity>
<xpdl:Activity Id="7" Name="Dopravit?">
  <xpdl:Description></xpdl:Description>
  <xpdl:StartMode>Automatic</xpdl:StartMode>
  <xpdl:FinishMode>Automatic</xpdl:FinishMode>
  <xpdl:Performer>automat</xpdl:Performer>
  <xpdl:ExtendedAttributes>
    <xpdl:ExtendedAttribute Name="Position">
      <xpdl:xPosition>190</xpdl:xPosition>
      <xpdl:yPosition>263</xpdl:yPosition>
    </xpdl:ExtendedAttribute>
  </xpdl:ExtendedAttributes>
</xpdl:Activity>
<xpdl:Activity Id="8" Name="Doprava k">
  <xpdl:Description></xpdl:Description>
  <xpdl:StartMode>Manual</xpdl:StartMode>
  <xpdl:FinishMode>Manual</xpdl:FinishMode>
  <xpdl:Performer>dopravce</xpdl:Performer>
  <xpdl:ExtendedAttributes>
    <xpdl:ExtendedAttribute Name="Position">
      <xpdl:xPosition>55</xpdl:xPosition>
```

```
              <xpdl:yPosition>238</xpdl:yPosition>
            </xpdl:ExtendedAttribute>
          </xpdl:ExtendedAttributes>
        </xpdl:Activity>
      </xpdl:Activities>
      <xpdl:Transitions>
        <xpdl:Transition From="0" Id="0" Name="" To="1">
          <xpdl:Description></xpdl:Description>
          <xpdl:Condition Type="CONDITION">[zakaznik] != NULL AND [popis od
zakaznika] != NULL AND [produkt] != NULL</xpdl:Condition>
        </xpdl:Transition>
        <xpdl:Transition From="2" Id="1" Name="" To="4">
          <xpdl:Description></xpdl:Description>
          <xpdl:Condition Type="CONDITION">[nutno videt] = NE</xpdl:Condition>
        </xpdl:Transition>
        <xpdl:Transition From="2" Id="2" Name="" To="3">
          <xpdl:Description></xpdl:Description>
          <xpdl:Condition Type="CONDITION">[nutno videt] = ANO</xpdl:Condition>
        </xpdl:Transition>
        <xpdl:Transition From="3" Id="3" Name="" To="4">
          <xpdl:Description></xpdl:Description>
          <xpdl:Condition Type="CONDITION">[doruceno od] = ANO AND [adresa
zakaznika] != NULL</xpdl:Condition>
        </xpdl:Transition>
        <xpdl:Transition From="1" Id="4" Name="" To="2">
          <xpdl:Description></xpdl:Description>
          <xpdl:Condition Type="CONDITION">[nutno videt] != NULL</xpdl:Condition>
        </xpdl:Transition>
        <xpdl:Transition From="7" Id="5" Name="" To="6">
          <xpdl:Description></xpdl:Description>
          <xpdl:Condition Type="CONDITION">[nutno videt] = NE</xpdl:Condition>
        </xpdl:Transition>
        <xpdl:Transition From="4" Id="6" Name="" To="5">
          <xpdl:Description></xpdl:Description>
          <xpdl:Condition Type="CONDITION">[vyreseno] = NE</xpdl:Condition>
        </xpdl:Transition>
        <xpdl:Transition From="4" Id="7" Name="" To="7">
          <xpdl:Description></xpdl:Description>
          <xpdl:Condition Type="CONDITION">[vyreseno] = ANO AND [popis reseni] !=
NULL</xpdl:Condition>
        </xpdl:Transition>
        <xpdl:Transition From="5" Id="8" Name="" To="8">
          <xpdl:Description></xpdl:Description>
        </xpdl:Transition>
        <xpdl:Transition From="8" Id="9" Name="" To="6">
          <xpdl:Description></xpdl:Description>
          <xpdl:Condition Type="CONDITION">[doruceno k] = ANO</xpdl:Condition>
        </xpdl:Transition>
        <xpdl:Transition From="7" Id="10" Name="" To="8">
          <xpdl:Description></xpdl:Description>
          <xpdl:Condition Type="CONDITION">[nutno videt] = ANO AND [adresa
zakaznika] != NULL</xpdl:Condition>
        </xpdl:Transition>
      </xpdl:Transitions>
    </xpdl:WorkflowProcess>
  </xpdl:WorkflowProcesses>
  <xpdl:Participants>
    <xpdl:Participant Id="0" Name="default">
      <xpdl:ParticipantType Type="HUMAN" />
    </xpdl:Participant>
    <xpdl:Participant Id="1" Name="operator">
      <xpdl:ParticipantType Type="ROLE" />
    </xpdl:Participant>
    <xpdl:Participant Id="2" Name="technik">
      <xpdl:ParticipantType Type="ROLE" />
```

```
    </xpdl:Participant>
    <xpdl:Participant Id="3" Name="automat">
      <xpdl:ParticipantType Type="SYSTEM" />
    </xpdl:Participant>
    <xpdl:Participant Id="4" Name="dopravce">
      <xpdl:ParticipantType Type="ROLE" />
    </xpdl:Participant>
  </xpdl:Participants>
</xpdl:Package>
```