Masaryk University

Faculty of Informatics

Procesní modelování operací s daty

Diplomová práce

Jan Pacner

Brno, podzim 2015

Prohlášení

Prohlašuji, že jsem Diplomovou práci zpracoval samostatně a použil jen prameny uvedené v seznamu literatury.

Bc. Jan Pacner

Vedoucí práce: RNDr. Jaroslav Ráček, Ph.D.

Poděkování

Děkuji panu Jaroslavu Ráčkovi, vedoucímu této diplomové práce, za entuziastický přístup k vedené této práce a hodnotné připomínkování, paní Táňe Kadlecové, ředitelce Montessori mateřské a základní školy Mozaika, za ochotu poskytnout reálná data o výuce a požadavcích MŠMT, tvůrcům modelovací aplikace bpmn.io za ochotu při řešení technických problémů a mé rodině za vycházení vstříc při psaní této práce, zejména pak nejmladšímu bratrovi Pavlovi za shovívavost při odmítání žádostí o pomoc s výukou.

Abstrakt

Diplomová práce pojednává o problematice modelování procesů a modelování dat a poukazuje na nedostatečné možnosti propojení těchto dvou historicky poměrně nezávislých směrů modelování systémů. V práci jsou nadefinovány moderní požadavky na procesní a datové modely a tyto jsou využity při specifikaci nového modelu XPM 1.0 kombinujícího procesní a datové modelování. K tomuto účelu byl navržený zcela nový a čistě abstraktní rozšiřitelný datový model CSDDM 1.0 zachycující komplexním způsobem semi-strukturovaná data spolu s analytickými požadavky na implementaci modelovaného systému. A to tak, aby byl snadno použitelný ve spojení s procesními modely. Datový model CSDDM 1.0 je hlavním přínosem této práce a kombinovaný model XPM 1.0 ukázkou možností integrace s BPMN 2.0.

Abstract

This master's thesis discusses the issue of process modeling and data modeling and emphasizes insufficient options of connecting these two historically rather independent directions of systems modeling. In the thesis there are defined modern requirements for process and data models and these are used for specification of the new model XPM 1.0, which combines process and data modeling. For this purpose a totally new purely abstract extensible data model CSDDM 1.0 depicting semi-structured data in a complex way together with analytical requirements for the modeled system implementation was designed. It's designed so, that it can be easily used together with process models. The data model CSDDM 1.0 is the main contribution of this thesis and the combined model XPM 1.0 is a showcase of integration options with BPMN 2.0.

Klíčová slova

procesní modelování, datové modelování, případy použití, komplexní model, XPM 1.0, operace s daty, jazyk pro definici dat, CSDDM 1.0, grafická notace, BPMN 2.0, UML

Keywords

process modeling, data modeling, use-case, complex model, XPM 1.0, data operations, data definition language, CSDDM 1.0, graphical notation, BPMN 2.0, UML

Obsah

1 Úvod

Tato diplomová práce si klade za cíl seznámit zájemce s pohledem na reálný svět okolo nás očima pozorovatele, který se snaží co nejrychleji pochopit základni aspekty a principy fungování činnosti v daném celku či oblasti, ať již se jedná o skupinovou činnost (organizovanou či neorganizovanou), činnost jednotlivce (člověka, živočicha, stroje) nebo činnost několika různorodých skupin dohromady.

Takovýto pohled na skutečnost má některá specifika, mezi nimiž nalezneme především pevně zvolenou míru abstrakce ve smyslu vhodného výběru předkládaných informací tak, aby pozorovatel nebyl zahlcen nepříliš podstatnými detaily, ale zároveň měl stále možnost tyto detaily kdykoliv rozkrýt. Neformální detaily tvoří často podstatnou součást vykonávané činnosti a jejich zanedbávání či dokonce opomíjení vede k nevoli v řadách účastníků společenství stejně jako naopak příliš holistický přístup, který detaily úmyslně opomíjí <citace>.

K tomuto účelu srozumitelného abstrahování se hojně využívá modelování, které je v tomto kontextu výrazně omezeno na úsporné zachycování nejdůležitějších aspektů reality ve vysoce srozumitelné a intuitivní podobě pro vnímání člověka. Protože vnější vnímání zdravého člověka je podmíněné dostupnými pěti smysly, využívá se nejsilnějšího smyslového vjemu, jímž je zrak, který tvoří přibližně 80% veškerého smyslového vnímání.

Proto se modely vytvářejí v grafické podobě a to ve dvoudimenzionální variantě (jiné dimenze se ukazují nepraktické – ať již nedostatečně přehledné pro laiky či naopak příliš komplikované i pro znalce). Důraz je kladen na intuitivní chápání a pro běžné použití (např. prohlížení) je vyžadována pouze minimální znalost celého modelovacího rámce a jeho omezení. Notace využívaná v těchto modelech se zpravidla skládá z triviálních geometrických obrazců a malého množství textu, který je využíván k neformálnímu popisování významu jednotlivých prvků notace v daném kontextu konkrétního modelu.

Aby bylo modelování univerzálně použitelné, je nutné zavést pravidla a omezení jak jednotlivé prvky modelovat a jak tyto spolu interagují. Jednotlivé prvky modelu jsou tedy formálně definované včetně jejich významu, chování a vztahu k ostatním prvkům. Tímto způsobem jsou vytvářeny nejen notace, nýbrž celé syntetické, avšak plnohodnotné jazyky, které musí být Turingovsky kompletní <citace>, aby bylo možné zachytit rozhodování se (viz. např. prvek brána v BPMN). Tyto jazyky lze tedy jednoznačně označit za programovací jazyky.

Model komplexní činnosti tedy zachycuje popis dílčích činností v jistém sledu a zároveň interakci mezi činnostmi, kde různé podčásti celého komplexního modelu jsou prováděny repetitivně. Tyto podčásti nazýváme procesy.

Modelování procesů lze nejen využít k zachycování aktuálního stavu existující komplexní činnosti, nýbrž i přesně opačně pro návrh ještě neexistujících komplexních činností (tedy včetně úpravy existujících modelů procesů). Z pozorovatele se tímto způsobem stává tvořitel. Od tohoto bodu je to již pouze krůček k pokročilým technikám vedení a řízení organizací, lidských aj. zdrojů. Právě toto je přesně problematika, která výrazně zajímá podnikovou sféru, kde je snaha zefektivnit komplexní činnosti, jejichž součásti mají být prováděny repetitivně.

Proto se v kontextu modelování procesů většinou bavíme o procesech podnikových a odtud plyne všeobecně uznávaný pojem „modelování podnikových procesů“. Samotný koncept je však zcela univerzální, jak je popsáno výše, a je výhodné ho využít i v ostatních oblastech.

Modely procesů se využívají již několik desítek let, ačkoliv historicky nejdříve byly používány výrazně omezené varianty a to pouze v několika málo oblastech (jednou z nich byl např. transport). Tyto modely postrádaly vyjadřovací schopnost např. pro rozhodování a byly tedy vhodné pouze pro zachycování zcela lineárních nerozvětvených komplexních činností (jednalo se tedy o pouhé seřazení činností do správného pořadí). V moderních modelech procesů (např. v notaci BPMN) se však setkáváme s notací pro tzv. rozhodování a větvení, což umožňuje modelovat jakýkoliv proces a způsobuje Turingovou úplnost.

Pro rozhodování je nutná existence kritéria, dle kterého se bude rozhodovat kterou větví se bude proces dále ubírat. Toto kritérium v podobě nějakých dat <citace> je jediný prvek Turingovsky kompletního procesního modelu, u kterého je nutné mít tato data dostupná nejen v modelu, ale i v reálné implementaci procesu, protože se na jejich základě bude vybírat větev kudy se bude proces dále ubírat. Toto kritérium může být též parametrizované a na základě aktuálních provozních dat se může měnit. Proto je nutné tato data reprezentující rozhodovací kritéria oddělit od modelu samotného.

Tato data reprezentující parametrizovaná rozhodovací kritéria lze měnit mimo modelované procesy (externě) a nebo během provádění dílčích činností v procesech (interně). Ukazuje se, že díky vysoké digitalizaci v podnikových prostředích jsou tato data měněna nejčastěji interně. V takovém případě je nanejvýš vhodné zachovat konzistenci mezi modelováním dílčích činností a rozhodovacích kritérií a je nutné formálně modelovat i práci s daty uvnitř činností. Zde narážíme na problém, že žádná z rozšířených, standardních modelovacích technik pro procesy neobsahuje prostředky pro modelování dat a práce s nimi. Přitom nejen rozhodovací kritéria vyžadují konzistentní modelování dat, nýbrž i mnohé modelované procesy jsou často do různé míry digitalizované, a v takových případech je nutné zachytit s jakými daty se v jednotlivých dílčích činnostech v procesech pracuje.

Procesní modelování tedy intuitivním způsobem řeší problematiku přehlednosti komplexních činností abstrakcí do formy procesů. Jakmile však začneme vytvářet větší množství procesních modelů, i tyto modely nám začnou připadat nepřehledné ve smyslu, že neodrážejí zcela holistický pohled na celý modelovaný systém, tedy na procesy samotné. Řešením se jeví seskupování dle vhodných kritérií. V modelovací notaci UML můžeme nalézt mj. i model případů užití, který velice dobře reflektuje potřebu zcela nejhrubějšího pohledu na celý modelovaný systém, protože pozorovatel se může ztotožnit s danou uživatelskou rolí systému a má okamžitý přehled, které dílčí činnosti ze kterých procesů může provádět. Prerekvizitou pro tvorbu případů užití je tedy definice přístupových práv k jednotlivým logickým celkům v datech, se kterými se v dílčích činnostech procesů

pracuje.

Tato práce demonstruje jak se některé modelovací nástroje a standardy procesních modelů vyrovnávají s nastíněnými problémy přehlednosti komplexního modelu, procesního a datového návrhu a předkládá nový, avšak se standardy kompatibilní způsob, jakým je možné tyto problémy konzistentně řešit. Je též zběžně rozebrána problematika modelování dat, a to i v souvislosti s nekonzistencí mezi datovým modelem a realizovaným datovým systémem (nejčastěji ve formě databází) a s tím spojenými limitacemi.

1.1 Služby a procesy

Dnešní systémy jsou často modelovány dle ekonomické představy o jakémkoliv systému, který vytváří nějakou hodnotu pro zákazníka. Tyto systémy pak nazýváme servisními systémy.

Servisní systém má definovanou specifikaci a můžeme na něho pohlížet jako na trojici entit poskytovatel, odběratel a výsledná hodnota. Přičemž vztahy mezi těmito entitami zajišťují, že jak poskytovatel, tak odběratel spolupracují na vytváření výsledné hodnoty.

Přičemž odběratel je ten, dle kterého se výsledná hodnota formuje a tedy jeho spolupráce má formu přirozeného projevování přání a poskytování co nejpřímější zpětné vazby poskytovateli, popř. výraznější spoluúčasti pokud to daná služba vyžaduje (např. pokud službou má být zážitek z tandemového seskoku padákem, pak bez fyzické spoluúčasti nelze službu využít, resp. nelze dosáhnout nabízeného zážitku). Poskytovatel je naopak ten, který nabízí službu, jejímž výsledkem je polotovar, který je dotvořen spoluúčastí odběratele. Poskytovatel dělá vše pro to, aby zásahy a vstupy odběratele byly co nejobsáhlejší co se relevantnosti týče, avšak jejich zprostředkování bylo pro odběratele dostatečně nenáročné.

Každá poskytovaná služba má tedy minimální životní cyklus, ve kterém figuruje interakce mezi poskytovatelem a odběratelem služby. Na obou stranách probíhá vyhodnocení přijatých informací a následující reakce. Tyto jednotlivé části celého cyklu můžeme chápat jako procesy, které mezi sebou komunikují. Dokonce lze dokázat, že naprosto všechny servisní systémy dle výše uvedené definice trojicí entit, jsou plně popsatelné procesními modely.

Tato skutečnost se dá aplikovat zpětně a každou podmnožinu libovolného procesního systému, ve které lze nalézt identické zobrazení trojice entit servisního systému, lze nazvat službou. Právě tato aplikace je velice rozšířená v oblasti IT a modelování jakýchkoliv systémů. Protože se tím často ztrácí přehled o tom, co vlastně služba nabízí, může být tento rozšířený pohled na služby matoucí. Zvlášť když v IT tato definice servisního systému jako souboru procesů splňujících jednoduchá pravidla,

často nedosáhne tak blízko k odběrateli služby jako služby hmatatelnějšího charakteru.

Pozitivním dopadem tohoto pojetí servisních systémů je možnost využít standardní prověřené metody analýzy, implementace a nasazení procesů. Můžeme tedy prohlásit, že procesy pohání ať již v jakékoliv formě všechny servisní systémy a tedy všechny služby, ve kterých odběratel nefiguruje jako pasivní entita.

Při vytváření a zavádění nové služby můžeme tedy metodou shora dolů vhodným dekompozičním postupem zachytit jakoukoliv činnost, ať již lidskou či strojovou či pouze imaginární souborem procesů, které se stávají nezbytným prostředkem pro realizaci a fungování služby. Tento pohled na proces nazýváme Process as a Service po vzoru pojmu Business Process as a Service používáného firmou IBM, což je úzce vymezená varianta použití procesního přístupu k realizaci služeb.

2 Modelování procesů

V případě, že budeme uvažovat o jakékoliv službě, činnosti, pohybu či změně a budeme tyto potřebovat replikovat, tedy zopakovat jejich podstatu (např. znovu provést danou činnost za daných podmínek) nebo je porovnávat mezi sebou, je nutné nejprve nalézt a definovat abstrakci, ve které bude možné zachytit jejich všechny potřebné aspekty a to v minimálním možném tvaru.

Základním způsobem je slovní popis ve vybraném lidském jazyce. Tento způsob popisu procesů však nesplňuje výše uvedený požadavek na minimální možný tvar. Přesto se stále jedná o nejrozšířenější variantu popisu procesů, ke které není nutné mít žádné další znalosti a hodí se tedy pro prostředí, ve kterém jsou procesy poměrně krátké a kde se očekává provádění procesů přímo lidským zdrojem.

Ukazuje se však, že mnohé lidské úkony lze do jisté míry automatizovat a přenechat je stroji. V takovém případě však popis lidským jazykem nevyhovuje, protože je příliš vágní a stroj není schopen vybrat správný význam daného zápisu. Obdobný problém nastává i v případě popisu rozsáhlých a rozvětvených procesů čtených či prováděných člověkem.

Vznikla tedy snaha formalizovat zápis procesů s cílem sjednotit požadavky na popis procesů a zajistit celosvětovou standardizaci nalezených typů a atributů. Vedlejší ideou, která se však ukazuje jako ještě důležitější, je přesun z lineárního pojetí popisu procesu do planárního pojetí popisu.

Tato nově přidaná dimenze umožňuje využití geometrického vnímání člověka, vhodnější využití barev pro zvýšení názornosti a výrazně přispívá k celkové přehlednosti i složitějších modelů. V neposlední řadě je kladen důraz na minimalizaci množství textu, protože pro člověka je náročné vytvořit si mentální abstraktní model textově popsaných skutečností. Výrazně jednodušší na pochopení je vnímání obrázkového znázornění. Modely se tedy snaží oprostit od lineární reprezentace a pojmout model ve 2D, aniž by se nechaly ovlivnit původní linearitou.

První modely se nesly v duchu existujících matematických konstrukcí v čele

s teorií grafů. Tyto ad-hoc počiny byly poté nahrazovány velice silným formalismem Petriho sítcí a nakonec rozmělňovány a obohacovány o specifické prvký z vyšších úrovní abstrakce, aby se předcházelo opakujícím se vzorům v modelech. Vznikly různé grafické notace a u všech lze nalézt silnou snahu o co nejlepší vizuální znázornění, které je prezentovatelné i laikům, protože původní primární využití těchto nástrojů bylo k pasivnímu popisu situace (např. pro pochopení fungování firmy) a až později se objevily snahy o optimalizaci a jako poslední se začaly notace využívat k návrhu prozatím neexistujících procesů.

Základním principem je určení začátku a konce procesu (tj. vyjasnění jak vypadá počáteční stav před spuštěním procesu a jak vypadá koncový stav), rozdělení činností v procesu na logické celky a jejich uspořádání za využití přechodů a specifikace podmínek pro přechody.

Hlavní limitací modelování procesů je neúprosná diskretizace celého prostředí. Tedy problematika rozdělování činnosti na zcela oddělené celky (ať již jsou těmito celky samotné procesy či soubor dílčích činností v procesech). Diskretizace klade vysoké nároky na schopnosti tvůrce modelu systému a po naimplementování procesu působí na člověka často velice strojeně až nepříjemně. Diskretizace však umožňuje naprosto bezproblémové určení co bylo započato a co bylo dokončeno, což jsou jedny ze základních vlastností pro efektivní provádění jakékoliv činnosti.

Jednou z nadstandardních možností, které nám modely procesů přináší jsou diskrétní simulace procesů, jejich statistické vyhodnocení a následná úprava procesu dle nalezených úzkých hrdel. Pro tyto účely lze použít základní metodický postup dle následujících typů optimalizací procesů.

- Continuous Process Improvement (CPI) – změna v množství a kvalitě zdrojů (v modelu procesu nedochází k žádným změnám)
- Business Process Reengineering (BPR) – změna uvnitř vybraného procesu (např. slučování, rozpad a přesunování jednotlivých činností, paralelizace, …)
- Business Process Redesign (BPRD) – kompletní předělání procesů (mohou být změněny i vstupy a výstupy procesů)

Další významnou roli hrají modely procesů při predikování vlastností neexistujících služeb a činností. Toho se hojně využívá např. při dimenzování kapacit pro výrobní postupy, logistiku, odbavovací terminály, a mnohé další.

Nejmodernějším použitím modelů procesů je jejich obohacení o abstrahované implementační detaily (např. vyhození výjimky) a následné spuštění modelu v některém podporovaném BPM Engine. Další možností je poté kompilace modelu do množiny spustitelných programů a to platformě nezávisle.

2.1 Proces

Proces v kontextu této práce je chápán jako po částech uspořádaná množina činností, které jsou opakovatelné. Činnost je jakákoliv akce proveditelná člověkem nebo systémem samostatně (automaticky).

Definic procesu existuje mnoho, ale všechny se snaží zachytit následujících atributy.

1. Totální rozpad libovolné akce měnící stav systému do logických celků.
2. Jednoznačné určení pořadí těchto logických celků (avšak nejen vůči přímo sousedícím logickým celkům).
3. Omezení způsobu rozpadu tak, aby provádění výsledných logických celků bylo opakovatelné.

Historicky se namísto slova proces používal pojem „workflow“, který především vycházel z představy zřetězení mechanických činností ve výrobnách či mezi výrobnami a „toku“ hmatatelných částí výrobků mezi nimi. Původní význam se tedy soustředil na nízkoúrovňové chápání procesů probíhajících přímo v těsném kontaktu s lidským činitelem. Důraz byl kladen na logistiku (ať již mezi stroji a lidskými pracovníky uvnitř výrobny či mezi výrobnami).

Postupem času se představa logistiky začala aplikovat i na malé logistické úkony jako je přenášení papírů mezi odděleními v rámci administrativních

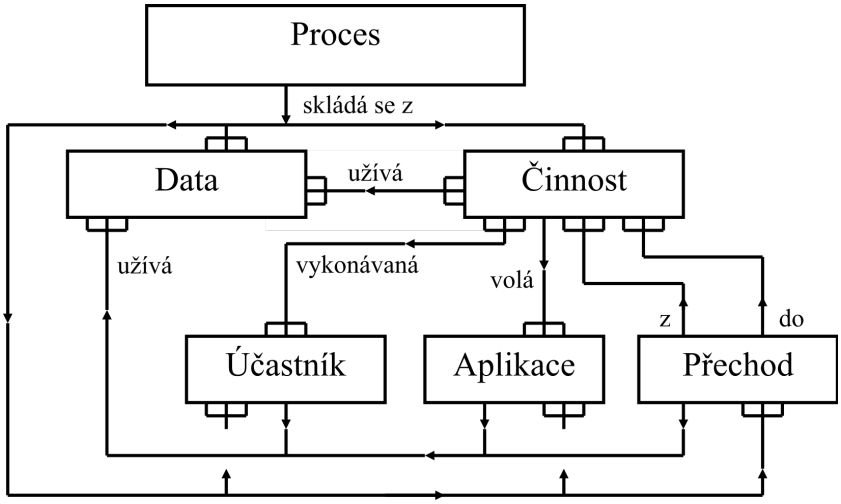
činností. Tyto činnosti však vyžadovaly změnu uvažování o procesech jako lineárním sledu aktivit přesunutí se k procesu jako libovolnému orientovanému grafu činností.

Vzhledem k jednoduše měřitelné užitečnosti navržení a řízení pomocí procesů se tyto velice rychle rozšířily (zejména do velkých organizací), aniž by byl brán ohled na pracovníky, kteří dle těchto procesů prováděli činnosti. Tento ohled se často týká detailů, které jsou při návrhu procesů opomíjeny a přitom tyto detaily dávají pracovníkům volnost a možnost chápat proces jako doporučený způsob jak pracovat s garancí, že neudělají chybu. Pracovníci zpravidla nemají ani možnost jakkoliv navržené procesy upravit ani jinak ovlivnit.

Kvalita navržených a implementovaných procesů se velice různí a globálně je spíše nižší. Z tohoto důvodu je slovo proces mezi pracovníky často chápáno negativně. Navzdory negativním konotacím je procesní přístup k řízení práce a organizace vysoce efektivním nástrojem.

Přílišná efektivita je však nepřirozená, a tak je nutné při modelování brát zřetel i na lidské činitele, kteří se budou procesu účastnit, aby nedocházelo k výše zmíněným negativním dopadům. Tato hrozba je jednou z nejsilnějších limitací procesního pojetí. Zejména proto, že je procesům vlastní a neodstranitelná. Závisí tedy pouze na návrháři procesu, zdali tuto limitaci zmírní či nikoliv.

Dnes se nejčastěji setkáváme s tzv. podnikovými procesy. Je to z důvodu, že právě podniky, zejména ty velké, provádějí velké množství kancelářské práce (vykazování, žádosti, smlouvy, atd. se provádí za pomocí dokumentů), která sama o sobě není přínosná a výsledné benefity neodpovídají náročnosti na zdroje. Tato činnost je velice repetitivní a tedy se jedná o zcela jasný procesní postup. A takovéto rutinní činnosti lze pomocí procesního modelování výrazně zefektivnit (viz. metody v úvodu této kapitoly) či do vysoké míry automatizovat a zrychlit.



Obrázek 1: Metamodel procesu [25]

2.2 Modelovací notace a jejich vlastnosti

Vývoj modelovacích notací směřuje směrem sjednocování důležitosti jednotlivých aktivit[1], čímž je vytvářena fixní úrovně abstrakce. Například aktivitu znázorňující logistiku je nutné modelovat jako běžný úkol s atributem zachycujícím přinejmenším minimální trvání a nikoliv speciální grafický prvek jako tomu bylo v modelech vytvářených ekonomy a logistiky (např. Flow Process Chart [26]).

Toto řešení fixní úrovně abstrakce se ukazuje výhodné jak pře přehlednost celého diagramu, tak pro to, že pro každou zainteresovanou stranu pracující s modelem je důležité něco jiného (např. pro analytika jsou nejdůležitější aktivity, které budou vyžadovat více práce programátora, kdežto pro manažera logistiky bude ve stejném modelu nejdůležitější aktivita převozu zboží skrze celou Evropu).

Nedílnou součástí modelů je jejich vizuální reprezentace, a proto jsou k nejpoužívanějším modelům uvedeny ukázky grafické notace v případě, že model tuto notaci poskytuje.

1 Aktivita je popis logického celku práce, která má být provedena (ať již manuálně člověkem či automaticky systémem).

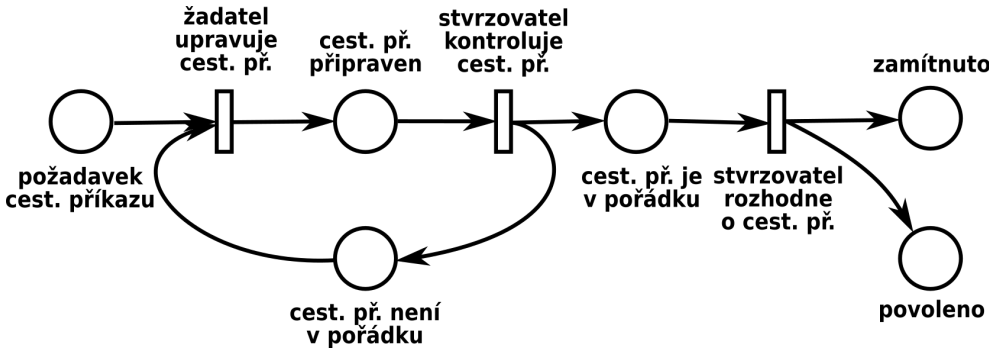
Ná následujícím případovém modelu budou ukázány nejdůležitější vlastností demonstrovaných modelů. Případový model popisuje situaci, ve které figurují dvě role – žadatel a stvrzovatel. Žadatel žádá stvrzovatele o schválení cestovního příkazu a stvrzovatel má možnost žádosti vyhovět a zaslat ji k dalšímu automatickému zpracování, žádost zamítnout s vysvětlením či žádost vrátit žadateli zpět s požadavkem na úpravu dílčích součástí žádosti.

Tento jednoduchý případ se snaží zachytit několik podstatných vlastností procesních modelů. A sice nalezneme v něm následující.

- Započetí a ukončení procesu.
- Předávání řízení toku procesu mezi více rolemi.
- Rozhodovací podmínku, určující kterou činností se bude dále pokračovat.
- Smyčku v řízení toku procesu.
- Manuální činnosti.

2.2.1 Petriho sítě

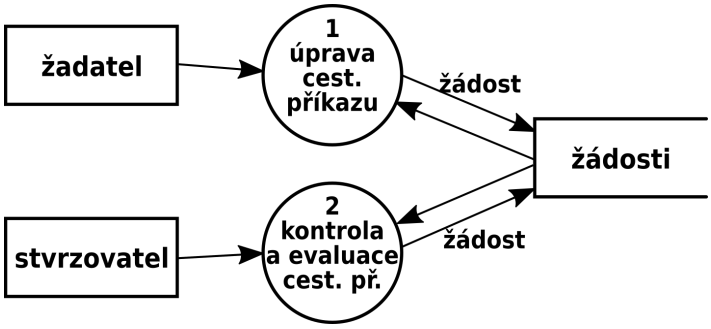
Tento rigorózní modelovací princip byl představen v disertační práci Kommunikation mit Automaten [15]. Model je schopen zachytit nedeterminismus na rozdíl od zdánlivě podobného modelu konečného automatu. Petriho síť se proto dá využít pro modelování jakýchkoliv diskrétních problémů reálného světa (např. fungování pokladny, ke které přicházejí zákazníci nebo letový provoz na mezinárodním letišti) a tedy i procesů. Petriho síť však neposkytuje žádnou abstrakci uživatelských rolí, a tak je nutné tyto informace dopsat formou slovních poznámek či použít Petriho síť např. s plaveckými dráhami ad-hoc.



Obrázek 2: Petriho síť - proces schvalování cestovního příkazu

2.2.2 Data Flow Diagram

DFD [8] zachycuje práci s daty jako sítí relačních členů. Nejedná se však o zcela procesní notaci, protože nesplňuje druhou podmínku o pořadí logických celků činností. DFD obsahuje prvky, které přímo určují role (tzv. terminátory), datové úložiště a základní datové operace (vytvoření, modifikace, odstranění), procesy (bezestavové funkce) a tok dat mezi těmito všemi prvky.

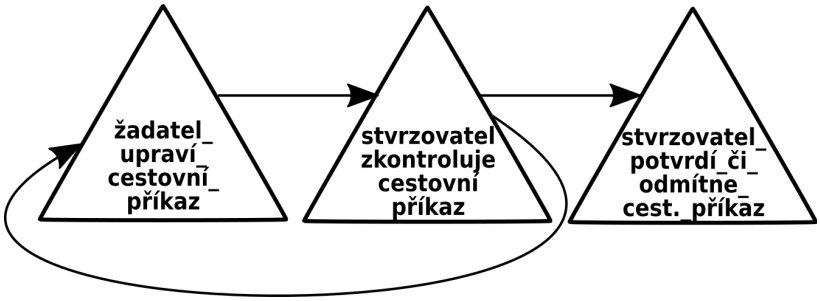


Obrázek 3: DFD - proces schvalování cestovního příkazu

2.2.3 Ad-hoc notace

Ad-hoc notace jsou používané v mnohých starších výzkumných článcích. Např. SEAM [1], kde je využito zcela vlastní sady grafických elementů.

SEAM rozlišuje pouze dva typy činností – automatické a manuální. Toto rozdělení lze nalézt i u ostatních notací, protože se jedná o nezbytnou vlastnost pro modelování procesů.

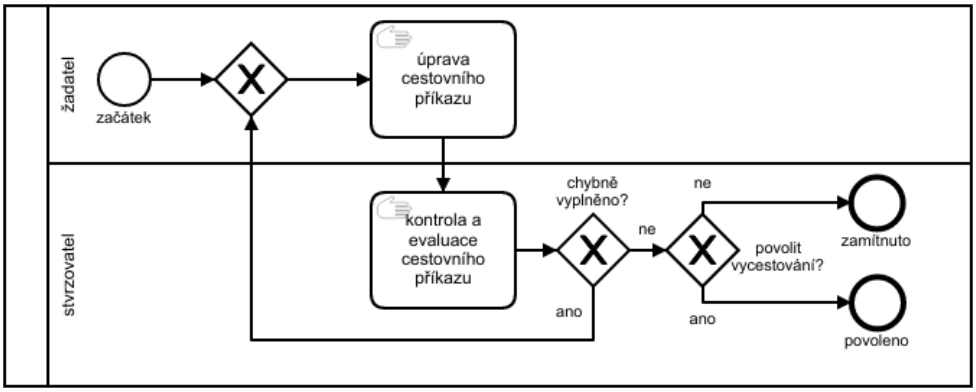


Obrázek 4: SEAM - proces schvalování cestovního příkazu

2.2.4 BPMN

Moderní notace vycházející z pozorování, že příliš formální definice v business světě nenacházejí pochopení a je tedy mnohem výhodnější soustředit se raději na inženýrské chápání jako např. Flow process chart [26]. BPMN 2.0 bylo vytvářeno v souladu s aktuálními trendy ve světě podnikových systémů. Tyto systémy se vyznačuji převažujícím jazykem implementace Java a databázovým jazykem SQL, avšak tvůrci si správně uvědomili, že stávající popisování dat je v podnikových systémech výrazně nevyhovující a proto ze specifikací pro procesní modelování zcela odstranili datové modelování. Toto je doplněno v XPM 1.0 (viz. kap. Kombinované modely).

Hlavními prvky BPMN 2.0 jsou činnosti (manuální či automatické), bazén s plaveckými dráhami (znázorňující uživatelské role), události (např. začátek procesu, časovač, zasílání zpráv), brány (rozdělují či slučují tok v procesu dle vyhodnocení v podmínky), spojovací šipky či úsečky (naznačují pořadí v toku procesu) a doplňkové objekty (např. anotace či datový dokument).



Obrázek 5: BPMN 2.0 - proces schvalování cestovního příkazu

2.3 Limitace notací a možná řešení

Notace, které mají tendenci modelovat procesy nízkoúrovňově (např. Petriho sítě) nemají většinou žádné známé limitace. Je to dáno tím, že jsou buď identickým zobrazením nějaké matematické konstrukce a nebo se od ní přinejmenším nevzdalují a zachovávají sémantiku a někdy i část notace. Naopak notace snažící se poskytnout analytikovi mnohé vyšší koncepty a vyhýbat se nízkoúrovňovému pojetí mají již z definice více „zakázaných kombinací“ a tyto způsobují obtíže.

Např. v BPMN 2.0 se standardizovanou sémantikou nelze zachytit poměrně častý případ užití ve výrobě. A sice zamezení náhodného přidělování činností jednotlivým zdrojům majícím stejnou roli a pracujícím v různých instancích stejného procesu. Např. dvě stejné výrobní linky, každá mající dvě oddělené činnosti (aktivity), které vyrábějí to samé paralelně a každou linku obsluhuje právě jeden zaměstnanec, přičemž oba zaměstnanci vystupují ve stejné roli. V BPMN 2.0 nelze zachytit, že jeden zaměstnanec bude mít na starosti vždy právě celou jednu linku. Lze zachytit pouze jednotlivé oddělené činnosti ze kterékoliv ze dvou linek. Tím však říkám, že zaměstnanci mohou libovolně (náhodně) mezi činnostmi na obou linkách přebíhat v případě, že budou mít zrovna malou vytíženost své linky. Přebíhání zaměstnanců mezi linkami však nebyl můj záměr.

K tomuto problému nalezneme dvě řešení:

1. Zkopírovat proces linky a pro každý z nich vytvořit unikátní roli.

   Výhody: lze využít stávající simulační nástroje. Problematika je jasnější čtenáři modelu.

   Nevýhody: nesnadná údržba obou procesů a obou rolí a tedy výrazně vyšší riziko nekonzistentně a tedy nesprávně nastavených vlastností procesů. Nemožnost jednoduše namodelovat data pro kopie procesů (je nutné všude vytvořit reference do původního datového modelu v kopírovaném procesu).

2. Rozšířit sémantiku BPMN 2.0 a využít např. Group s opačnou definicí: Group ovlivňuje tok v procesu tak, že aktivity v Group nemohou být vykonány žádnou jinou instancí dané role než tou, která provedla první aktivitu v Group.

   Výhody: je to přesně ono, čeho jsme chtěli docílit.

   Nevýhody: zásadním způsobem nezachovávám standardní sémantiku BPMN a v případě zmizení instance obsluhující role v průběhu procesu zůstane tento navždy zaseknutý.

Další nevýhodou vizuálních procesních modelů je velice špatná podpora pro znovupoužití částí procesů. V komplikovaných procesech se začínají objevovat opakující se úseky, které mají být prováděny různými rolemi v různých časech. Kopírování na více než jedno místo v takovém případě zcela znehodnotí přidanou hodnotu vizualizace. Vytvoření podprocesu příliš nepomůže, protože místo pro něj bude vhodné stále pouze pro jeden proces. Vytvoření plnohodnotného procesu však znamená horší udržovatelnost (desítky krátkých podprocesů modelovaných jako velké procesy není šetření práce ani zachování vizuální srozumitelnosti).

Všechny uvedené procesní notace, jakožto nejvýznamější a nejrozšířenější notace zachycují chování. Ve všech však zcela chybí popis s čím se pracuje. Procesy však vždy pracují s daty. Specifikace dat však v těchto modelech chybí. Někdy není nutné přesně vědět se kterými daty se pracuje (postačuje vágní popis „smlouva“ či „mapa s vyznačenou cestou“). V mnoha případech však tento neformální popis dat nedostačuje.

Data je tedy nutné zahrnout do modelu. Procesní model však již využívá většinu 2D prostoru a je tedy nutné datový model uskromnit či využít překryvu. Více o této problematice v kapitolách o kombinovaných modelech.

3 Modelování dat

V rané historii byly digitální informace zpracovávány počítači, které měly pevně danou hierarchii pamětí a programy se připravovaly pro konkrétní počítač tak, že si programátor důkladně nastudoval problematiku, hluboce se zamyslel a celé to naprogramoval. Velice brzy se však objevila potřeba řešit rozdíl mezi rychlostí externích perzistentních pamětí a rychlou pamětí pro výpočty. Objevily se první pevné rotační disky a efektivní práce s těmito disky vyžadovala plánování zápisů, přepisů, zarovnávání a další úkony potřebné k práci s daty.

Jak množství dat velice rychle rostlo, ukázalo se, že čistě programově je náročné vyhledávat a filtrovat data. Vznikaly proto pokusy o standardizaci strukturalizace dat a popisy těchto struktur byly prvními modely dat. Modelování dat se poté více oddělilo od popisu fyzické práce s daty do vlastního výzkumného proudu.

Mezi hlavní cíle modelování řadíme vizuální přehlednost a srozumitelnost. Modelování se však vždy potýká s problematikou výběru abstrakcí a množstvím dimenzí, které ještě v modelu zachycovat a které nikoliv. V případě dat se původní modely soustřeďovaly převážně na vztahy mezi daty a puntíčkářské zachycení malých atomických datových celků. To vše bez ohledu na provozní systémy, které tyto modely implementují, a tak model dat pro životně kritický systém a model dat pro knihovnický systém vypadaly vedle sebe naprosto totožně. Reálné požadavky však obsahují nutnost modelovat hlubší význam celého systému a jeho částí, v čemž původní modely zcela selhávají.

Modelování nám mj. umožňuje provádět studium systému ještě předtím, než je zrealizovaný. I výrazně abstrahovaný model poskytuje velké množství informací, pokud pomocí něho provedeme např. různé simulace. Vzhledem

k těmto možnostem modelů je modelování velice rozšířeným nástrojem v akademické sféře a v oblastech zabývajících se kritickými systémy.

Moderním trendem je práce s tzv. semi-strukturovanými daty, což jsou data, která si s sebou nesou anotaci co který úsek dat vlastně znamená a jak s ním pracovat a se kterými dalšími úseky dat souvisí. Jejich použití nalezneme ve veliké míře ve webových technologiích a stránkově- či formulářově-orientovaných aplikacích. Zajímavou vlastností těchto dat je, že je není nutné nijak serializovat, protože jsou již serializovaná. Toho se s výhodou využívá především při zasílání těchto dat přes síť, což je zdaleka nejrozšířenější způsob přenosu dat.

V případě semi-strukturovaných dat tedy téměř odpadá nutnost tato modelovat. Je však nutné zachytit alespoň vztahy mezi celou jednou semi-strukturovanou položkou a zbytkem strukturované databáze.

3.1 Požadavky

Vzhledem k náročnosti a rozmanitosti možností ukládání a poskytování dat bylo nutné specifikovat moderní požadavky na datové modely. Následující seznam požadavků zmiňuje i vlastnosti celkových systémů a nikoliv pouze modelů. Modely těmto systémům však předcházejí a pokud model danou vlastnost neumí zachytit, pak se zvyšuje riziko výběru nevhodné implementační technologie aniž by to bylo v modelu zmíněno a opodstatněno.

- Temporální vlastnosti (např. životnost či doba validity).
- Redundance (keš, pouze iniciální kopie, atd.).
- Nezávislost na technologii a na jazyku (ať již dotazovacím či programovacím).
- Velice dobrá podpora referencování (vhodné např. pro rekurzivní struktury).
- Existence formálního modelu na pozadí (díky kterému je možné provádět verifikace a simulace).

- Podpora semi-strukturovaných dat.
- Volitelné typování (zodpovědnost, zdali bude položka typovaná či nikoliv nese analytik) s podporou komplexních typů (včetně dynamických, tedy takových, které se mění za běhu systému).
- Podpora prostorových dat tak, aby bylo poté nad nimi možné provádět operace a dotazy jako: Je zvolený bod v ploše uloženého N-úhelníku?
- Podpora pro případné transakční zpracování (tento požadavek ovlivňuje sémantiku součástí modelu).
- Vizuální přehlednost (včetně požadavků na tvorbu identifikátorů).
- Podpora (v jakékoliv formě – např. pomocí referencí) pro kardinality jednotlivých vazeb mezi položkami.

3.2 Existující modely

Níže uvedené modely mohou zahrnovat i takové, které nejsou úplně modely, nýbrž pouze schémata. Schéma je zpravidla popis struktury semi-strukturovaných dat. Model je zpravidla nejvyšší abstrakce, kterou lze jakákoliv data (strukturovaná, semi-strukturovaná či nestrukturovaná) popisovat. Schéma též můžeme chápat jako speciální případ modelu, který je aplikovaný na menší rozsah semi-strukturovaných dat než obvykle.

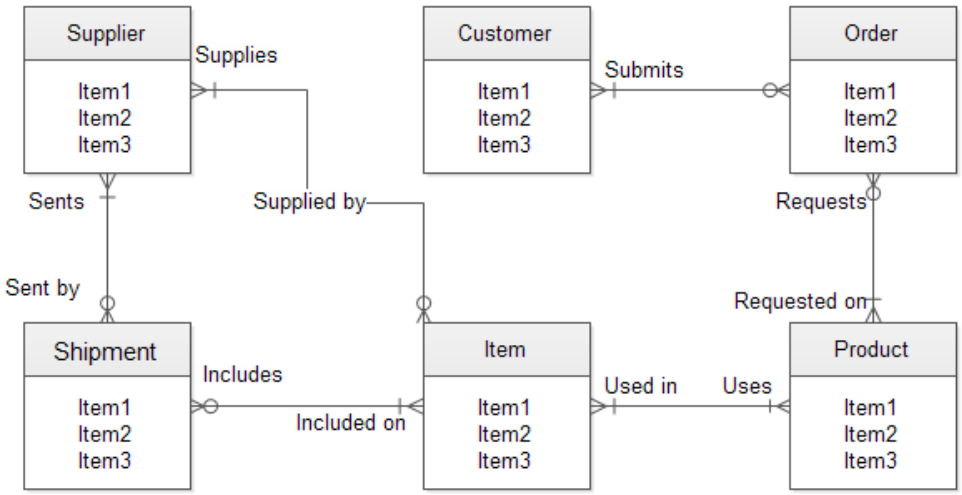
Uvedené příklady modelů nejsou záměrně sestaveny pro jeden univerzální příklad, protože se datové modely zásadním způsobem liší a v případě použití stejného příkladu by nevynikly důležité vlastnosti těchto modelů.

3.2.1 Relační model

Entitně-vztahový model (ERD), jehož finální podobu představil Peter Chen je nejrozšířenější metodou modelování dat. Umožňuje jak abstraktní modelování, tak fyzické. Cílem ERD je data neduplikovat a mít je všechny formálně popsané a kontrolované pomocí omezení. ERD sestává z tabulek definujících název a atributy (názvy a typy sloupců pro každou tabulku) a vztahových linií mezi nimi (tyto mohou mít slovní popisek o jakou vazbu

jde a mají kardinalitu na každé straně).

Pro ERD existuje i metodický postup, tzv. normalizace, pomocí kterého lze libovolný relační model přetvořit na ekvivalentní validní ERD vyhovující striktnějším požadavkům dané úrovně normální formy.



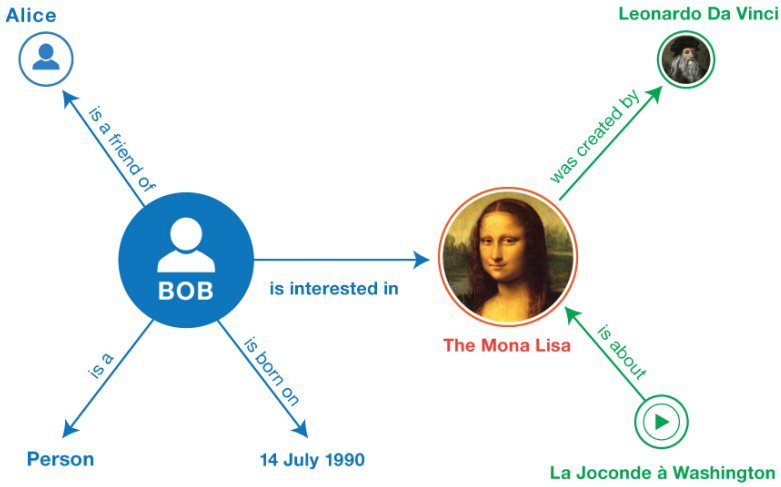
Obrázek 6: ERD - datový sklad pro distributora [27]

3.2.2 Funkcionální či logické modely

Tyto modely vycházejí nejčastěji z existujících programovacích jazyků, které zase vycházejí z elementárních matematických představ (např. logika). Do této skupiny řadíme funkcionální jazyky a logické jazyky. Zástupci těchto jazyků ukázaly nové přístupy jak lze provádět výpočty. Ve funkcionálních jazycích je kladen důraz na tzv. čistotu (absenci vedlejších efektů při výpočtu), v logických jazycích je kladen důraz na důkaz o platnosti formulí. Obě skupiny vyžadovaly definovat data a vyšší struktury jiným způsobem než bylo známé.

Z nových způsobů definice dat lze jako předního zástupce zmínit RDF a dále pak jakékoliv formáty postavené na stejném či velice podobném principu. RDF je trojice subjekt, predikát, objekt (podobnost s přirozeným jazykem není náhodná). RDF nemá definovanou syntaxi ani grafickou reprezentaci, a proto následující příklad je v pseudo kódu [38]:

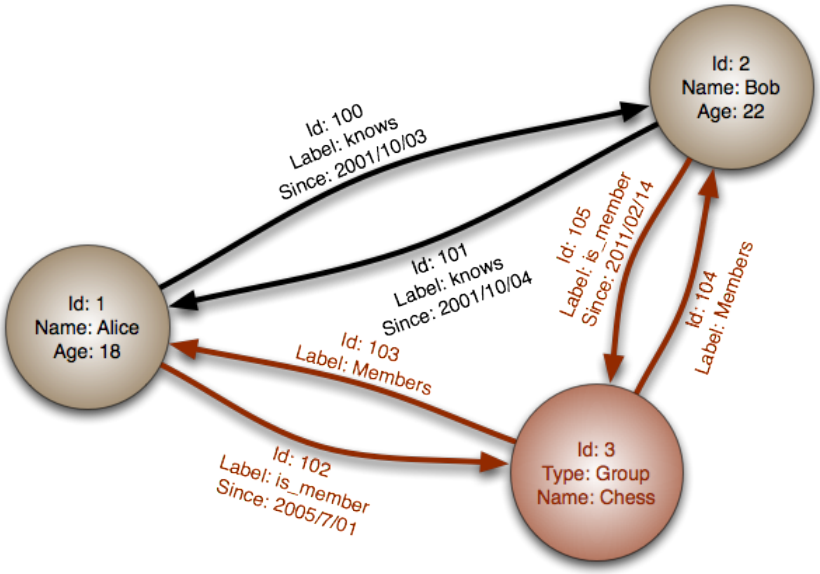
<Bob> <is a> <person>.
<Bob> <is a friend of> <Alice>.
<Bob> <is born on> <the 4th of July 1990>.
<Bob> <is interested in> <the Mona Lisa>.
<the Mona Lisa> <was created by> <Leonardo da Vinci>.
<the video 'La Joconde à Washington'> <is about> <the Mona Lisa>



Obrázek 7: Možná grafická reprezentace RDF [38]

3.2.3 Síťové a grafové modely

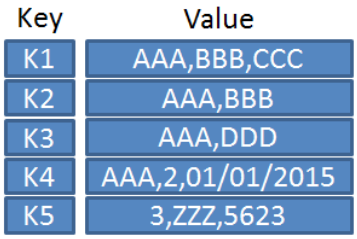
Další skupinou modelů jsou obecné síťové (grafové) modely dat využívané v moderních „graph-based“ databázích. Tyto modely nemají žádnou ustálenou konvenci notace ani syntaktické ani grafické. Vyznačují se vysokou mírou expresivnosti, avšak taktéž trpí problémem špatné pochopitelnosti, protože strukturování je často velice komplikované a nerozdělené na dostatečně separované logické celky. Zajímavostí je, že vzhledem ke genericitě těchto modelů lze s jejich pomocí simulovat jakýkoliv jiný datový model.



Obrázek 8: Vizualizace obecného grafového modelu [39]

3.2.4 Modely založené na asociativních polích

Jedním z neejednodušších možných datových modelů jsou modely využívající asociativní pole. Nejčastěji bývají nazývány key-value modely, což vychází z jejich jediné podstaty. Tou je schopnost k vstupnímu klíči vyhledat odpovídající právě jednu hodnotu (více klíčů však může vést ke stejné hodnotě). Abychom docílili možnosti modelovat i zanořená data, je nutné zajistit, abychom mohli využívat hodnoty jako skaláry (tedy přímo užitečná data) a nebo klíče (či vstupy pro výpočet klíčů), čímž zajistíme, že tímto způsobem získáme grafový či libovolný jiný model dat a tedy plnou sílu expresivity.



Obrázek 9: Model dat klíč-hodnota [40]

3.2.5 Dokumentově orientovaná schémata

Jedním z nejzajímavějších počinů na poli popisu dat a jejich struktury jsou schémata snažící se o rozšířenější a ucelenější popis než jednoduché modely zmíněné výše a zároveň se snaží přiblížit přirozenému chápání logických celků dat. Základní myšlenkou je dokument jakožto nejmenší datová jednotka, která však je schopná sémanticky zachytit mnoho informací (např. možné vizuální reprezentace, historii, vlastní strukturu, temporální vlastnosti, různé typy referencí, atd.). Vše ostatní je podřízeno této představě včetně způsobu vyhledávání, vizualizace, transformací atd.

Mezi nejstarší zástupce zařadíme SGML, které mělo však natolik volnou syntaxi a tak široké možnosti, že systémy s tímto formátem pracující byly příliš komplexní na to, jaké výhody tento formát přinášel. Postupem času se vyvinuly další formáty ať již založené na SGML či nově vytvořené, které nejsou tak rozsáhlé, ale přitom si zachovávají vysokou flexibilitu a umožňují stavbu efektivních a rychlých implementací. Mezi aktuálně nejrozšířenější patří XML Schema a JSON.

Obrázek 10: MongoDB JSON dokument s vnořenými poddokumenty [41]

3.3 Limitace, nekonzistence a možná řešení

Modelování přináší i rizika vyplývající z limitací založených na odebrání mnoha dimenzí reálného systému. Hlavním problémem je ztráta informace vyplývající ze součinnosti jednotlivých položek v modelu. V takovém případě se mohou vlastnosti, které jsou pro jednotlivé položky marginální, v celkovém systému ovlivnit zcela zásadním způsobem jeho chování. Jednou z těchto informací je např. očekávané množství instancí dané položky v reálném systému. Tato zdánlivě nedůležitá informace zásadně ovlivňuje např. v případě SQL jakékoliv složené dotazování.

Mezi další limitace, které se často objevují v datovém modelování můžeme zařadit následující.

- Závislost na jazyku (dotazovacím či přímo programovacím).
- Limitace formálního pozadí modelu (např. relační model pracuje s množinami, a tak je dodatečně přidané uspořádání drahou neefektivní operací, v relačním modelu je též vždy výsledkem operace relace, ale dost často je potřeba pouze jedna hodnota a vzniká nám zde tedy prostor pro problém duality výsledku operací, který se nepříjemně projevuje např. v SQL).

- Fixní výběr typů dat, což výrazně omezuje uživatele modelu a zamezuje holistickému pojetí. Nebo naopak rozhodnutí, že budou všechna data slabě typovaná, což výrazně negativně ovlivňuje výkon systému realizovaného dle takového modelu.
- Nedostatečná a neintuitivní práce s referencemi (např. model podporuje pouze dílčí část referencování – např. unikátní ID ke každé položce, avšak přenechává řešení konzistencí a práce s referencemi na uživateli modelu).
- Žádná či špatná podpora semi-strukturovaných dat.
- Chybějící temporální vlastnosti (např. unikátnost a životnost dat, využívání časových značek při pokročilých dotazech).
- Nedostatečná podpora pro prostorová data a operace s nimi.

Rozhodnout které z těchto limitací modelování by měly modely řešit a které ne není jednoznačné. Přesto lze prohlásit, že modelování by mělo umožnit vypořádání se s těmito limitacemi do takové míry, aby mohlo být z modelu jednoznačně určeno, které technologie lze pro systém využít a které nikoliv.

3.4 Nový datový model CSDDM 1.0

Na základě evaluace výše popsaných potřeb a limitací stávajících datových modelů byl v této práci navržen zcela nový datový model pojmenovaný Complex System Data Definition Model, dále jen *CSDDM*, ve verzi 1.0 .

Vlastnosti *CSDDM* jsou úmyslně připravené pro použití ve spojení s modely procesními a předpokládá využití práce s daty právě v procesech (ať již přímo v modelech či až naimplementovaných modelovaných systémech). Hlavním cílem CSDDM je přesná specifikace hranic pro implementaci práce s modelovanými daty v komplexních systémech. Komplexním systémem rozumíme systém, který s daty pracuje jak v případě uživatelských činností (např. BPMN Manual Task), tak systémových činností (např. BPMN Service Task) a kromě vztahů mezi daty popisuje i přináležející požadavky na vlastnosti implementace (např. výkonnost při zpracovávání modelovaných dat).

Specifikace hranic pro implementaci jsou nutné jak pro databáze využité k implementaci modelu, tak i přímo pro uživatelské rozhraní, jakožto výstupu vytvořeného právě na základě informací obsažených ve specifikaci hranic.

CSDDM tedy představuje vysoce parametrizovatelnou a rozšiřitelnou formální definici datového modelu a požadavků na implementaci. Verze 1.0 lapidární formou zachycuje semi-strukturovaná data [17] tak, aby bylo modelování metodou shora dolů maximálně usnadněno. Při modelování jsou tedy zcela úmyslně pominuty vlastnosti implementačních technologií, které jsou běžně používány pro daný druh modelovaného systému.

Tato kombinace umožňuje mj. předem, ještě ve fázi modelování, velice přesně určit chování reálného systému po implementaci. Dále tato kombinace garantuje, že mnohé důležité implementační parametry (např. která data budou v cache, která data jsou transakčně kritická, která data nevyžadují plnou konzistenci, atd.) nebudou nevhodně zvoleny z důvodu, že vybraná implementační technologie poskytuje právě pouze toto řešení. Omezení na základě nevhodných technologií jsou velice často natolik zásadní, že výsledný produkt neplní některé z předpokládaných funkcí. Přesné definování těchto požadavků již v první fázi vývojových projektů, tedy ve fázi analýzy, a to na úrovni detailu jednotlivých atomických datových položek, významným způsobem snižuje pravděpodobnost volby nevhodné implementační technologie.

Přesné definování však přináší zvýšené nároky na znalosti, schopnosti a čas analytika. Můžeme však použít analogii s programovacími jazyky, které řeší též problém práce s daty, avšak oproti datovému modelování a databázím jsou zaměřené více na volatilní pamět než na perzistentní. Vývoj programovacích jazyků se čím dále více zaměřuje na práci s daty tak, aby bylo možné ještě před spuštěním programu zachytit maximum potenciálních porušení konzistencí při běhu programu (např. jazyk Rust [18]). Tento přístup vychází z mnohaletých zkušeností, že v případě produkčních systémů je rychlejší, příjemnější a bezpečnější chybu odhalit ještě před nasazením systému než kdykoliv poté.

Vzhledem ke komplexnosti semi-strukturovaných dat již nelze modelovat

data metodou zespoda nahoru, tedy nejprve definovat atomické celky a poté z nich stavět složitější struktury, a proto je v *CSDDM* kladen důraz na metodu shora dolů. Tedy od nejvyšší abstrakce po nejnižší. Nejvyšší abstrakcí pro *CSDDM* bylo zvoleno uživatelské rozhraní (či ekvivalentně vysoce-úrovňové API), které je v dnešní době zdaleka nejčastějším výstupem jakéhokoliv produktu IT (pomineme-li plně autonomní systémy, vše ostatní se nějak vizualizuje, ačkoliv se nejedná o hlavní očekávané výstupy, nýbrž pouze např. o testovací rozhraní či konfigurační rozhraní).

Základním stavebním kamenem *CSDDM* je strom atributů. Atribut je atomická datová položka s několika standardizovanými, nastavitelnými vlastnostmi. Atomicitou atributu je myšleno, že systém implementující CSDDM hodnotu atributu (tj. datový obsah) dále nedělí a využívá ho jako jeden celek. Toto se poté promítá i do synchronizačních vlastností atributu a dalších parametrů. Zprvu tento přístup může silně připomínat dokumentově orientované databáze a modelování (např. MongoDB [19]), avšak vzhledem k vestavěné[2] podpoře referencí, definicím konzistencí, přístupových práv, nastavení jedinečnosti atd. se výrazně liší.

Z hlediska použití je jednou z nejzajímavějších vlastností CSDDM nezávislost na jakémkoliv datově orientovaném či programovacím jazyku či procesní notaci. Přesto model díky úzké vazbě na uživatelské či aplikační rozhraní zachycuje sémantiku nejjednoduššího typu dotazů (běžné získání dat) a jejich filtrování (např. pomocí referencí). V reálných systémech však tyto nejjednodušší dotazy a filtrování tvoří převážnou většinu všech dotazů, které se např. v informačních systémech a jiných uživatelsky orientovaných systémech využívají. Díky tomu lze použitím CSDDM pro takovéto systémy ušetřit nezanedbatelné množství zdrojů při realizaci těchto systémů.

Ostatní případy lze řešit součinností s procesním modelováním – např. pro efektivní vyhledávání dat by měl implementovaný procesní systém poskytovat programátorské rozhraní např. ve skriptovacím úkolu formou „handle“ s metodami umožňujícími deklarativním způsobem provést jakékoliv datové transformace s kterýmikoliv daty v systému dle práv, pod

2 „vestavěný“ v tomto kontextu označuje, že není nutné nic programovat ani zbytek modelu či dat na tuto skutečnost připravovat

kterými je daná činnost spuštěná. Poskytnuté metody mohou zahrnovat operace jako set, map, join (skrývající např. komplexitu Resilient Distributed Datasets [36], protože metody založené na orientovaných acyklických grafech či MapReduce [37] jsou příliš neefektivní) a pokud možno i introspekci.

ID	perm	uniq	constr	init	sync	consis
Collection00	add_rem	-	-	-	hardl...	write
Attr00	no_rw	none	none		hardl...	write
Attr01	write	none	int64	0	hardl...	write
Collection00	add_rem	-	-	-	hardl...	write
Collection01	add	-	-	-	hardl...	write
Attr00	read	no_procs_y...	string_utf8	?	hardl...	write
Attr01	read	no_procs_y...	string_utf8		hardl...	write
Collection02	rem	-	-	-	hardl...	write
Attr00	write	none	int64	1	hardl...	write
Collection03	add	-	-	-	hardl...	write
Attr00	write	none	string_asc...	x	hardl...	write
Attr01	no_rw	none	string_asc...	y	hardl...	write

Obrázek 11: Model semi-strukturovaných dat v CSDDM 1.0

3.4.1 Strukturální popis modelu CSDDM 1.0

Na pozadí demonstrované vizuální uživatelské reprezentace modelu CSDDM 1.0 nalezneme úplný strukturální popis modelu pro abstraktní modelový případ. Tento strukturální popis je vhodný zejména pro přesnou definici jednotlivých součástí modelu uvedených pod tabulkami.

Model je strukturovaný jak strom položek (uzly a listy), kde každá položka má vlastní množinu vlastností. Model je zobrazovaný ve formě tabulky pro lepší porozumění. Zobrazení tabulkou je též doporučené zobrazení pro studium modelu, protože neskrývá žádné detaily a vše je explicitní.

kind	*nest*	*id*	*perm*	*uniq*	*constr*	*init*	*sync*	*consis*
c	0	collection00	add_rem	-	-	-	hardlink	write
a	1	attr00	no_rw	none	none		hardlink	write
a	1	attr01	write	none	int64	0	hardlink	write
c	1	collection00	add_rem	-	-	-	hardlink	write
c	0	collection01	add	-	-	-	hardlink	write
a	1	attr00	read	no_procs_yes_roles	string_utf8	""	hardlink	write
a	1	attr01	read	no_procs_yes_roles	string_utf8	""	hardlink	write
c	1	collection02	rem	-	-	-	hardlink	write
a	2	attr00	write	none	int64	1	hardlink	write
c	1	collection03	add	-	-	-	hardlink	write
a	2	attr00	write	none	string_ascii_printable	""	hardlink	write
a	2	attr01	no_rw	none	string_ascii_printable	""	hardlink	write

Tabulka 1: Datový model úkolu T1 procesu P1

kind	*nest*	*id*	*perm*	*uniq*	*constr*	*init*	*sync*	*consis*
c	0	collection05	add_rem	-	-	-	hardlink	write
a	1	attr00	write	none	string_ascii_printable	""	hardlink	write
a	1	.collection00.attr01	write	none	int64	0	hardlink	write
a	1	.collection01.collection02.attr00	read	none	int64	0	hardlink	CRDT
c	1	.collection01.collection03	rem	-	-	-	hardlink	CRDT

Tabulka 2: Datový model úkolu T2 (odkazující obsah T1) stejného procesu P1

3.4.2 Jednotlivé součásti modelu CSDDM 1.0

Použitá terminologie.

- ***úkol*** je ohraničená práce, která se má udělat; v případě, že tato práce obsahuje data, tato jsou reprezentována uspořádanou množinou kolekcí a referencí
- ***kolekce*** je strukturální popis uspořádané množiny atributů
- ***atribut*** je atomická jednotka označující část dat, kterou lze chápat jako logický celek; ohraničení tohoto celku je definováno sadou omezení (viz. vlastnost ***constr*** níže)
- ***reference*** je schématický odkaz na existující záznam; její vlastnosti jsou totožné či kompatibilní s odkazovaným záznamem
- ***záznam*** je kolekce či atribut či reference

V následujícím seznamu jsou definovány jednotlivé vlastnosti modelu *CSDMM 1.0*. Demonstrace použití těchto vlastností lze nalézt např. ve strukturálním zachycení modelu výše.

1. ***kind***
   Význam: druh záznamu určující, zdali se jedná o kolekci či atribut.
   Povolené hodnoty: c , a
   Omezení:
   - Atribut musí splňovat $nest \geq 1$ (a tedy být v nějaké kolekci).

2. ***nest***
   Význam: hloubka zanoření záznamu ve jmenném prostoru úkolu.
   Povolené hodnoty: celá kladná čísla s nulou.
   Omezení:
   - Jedná se o pomocnou vlastnost, která není součástí modelu a slouží pouze pro zlepšení čitelnosti a pochopení CSDDM.

3. ***id***
   Význam: identifikátor reprezentující název atributu či kolekce.
   Povolené hodnoty: řetezec splňující regulární výraz *[_A-Za-z][_A-Za-z0-9]**
   Omezení:
   - Jsou rozlišována malá a velká písmena (X a x nejsou považovány za stejný znak).
   - Skládá se pouze z ASCII znaků (podpora UTF-8 chybí kvůli kompatibilitě).
   - Je globální pouze pro záznamy s $nest = 0$ (stejný identifikátor lze tedy použít v různých kolekcích bez mnohoznačnosti).
   - Pokud se nejedná o referenci, pak je identifikátor plně kompatibilní s SQL a URI.
   - Podporuje referencování existujícího schématu záznamu takto:
     - Referencování nic neříká o datech, které schéma popisuje, protože referencování pouze napodobuje referencované schéma záznamu (včetně jeho obsahu, obsahu jeho obsahu, atd.).
     - Syntaxe pro externí schémata záznamu:
       - Odpovídá URI ukazující na externí API kompatibilní s CSDDM.
       - URI musí začínat prefixem „<***schema***>**:**“.
       - Externí URI API musí podporovat předávání parametrů (přihlašovací údaje, nastavení pro získávání dat, bezpečnostní nastavení, atd.).
       - Externí schéma musí podporovat předání informací o množině omezení a dalších vlastnostech zprostředkovávaných položek (možné řešit introspekcí).
     - Syntaxe pro lokální schémata záznamu:
       - Shora dolů (směrem od kořene stromu) je použita tečka jako oddělovač v celé cestě k referencovanému schématu záznamu (plná cesta začíná ID procesu, které může být

vynechané v případě referencování procesu, ve kterém je tato plná cesta použitá, a pokračuje tečkou následovanou záznamem s $nest = 0$).

- Zespoda nahoru (směrem od stávajícího záznamu) je použita dvojtečka („dvě tečky“) jako reference na rodiče v relativní cestě k referencovanému schématu záznamu (pro referencování sebe sama lze použít „:nazevStavajicihoZaznamu.“).

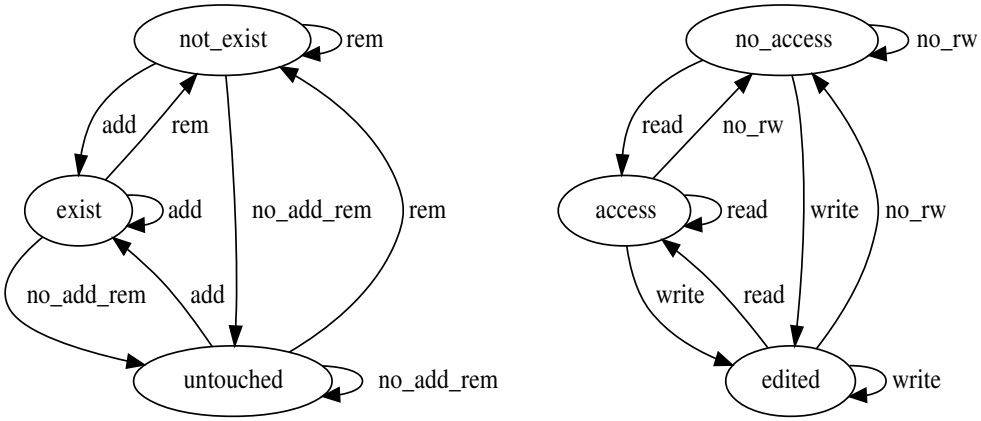
- Oba oddělovače segmentů (tečka a dvojtečka) mohou být libovolně kombinovány v jedné cestě.
- Reference obsahují nekompatibilní SQL znaky.
- Reference, která je kolekcí, nemůže mít žádné dodatečné potomky (ani atributy ani kolekce) ani z ní nemohou být žádné odebrány.
- Reference, která je atributem musí mít $nest \geq 1$ (a být tedy součástí kolekce).
- Všechny atributy reference, která je kolekcí (tyto atributy tedy nejsou viditelné) jsou vždy stejné jako atributy referencované kolekce (kromě *nest* , který je relativně přepočítaný a kromě *id* , který je relativně uříznutý a poté prefixovaný cestou k aktuální referenci).

- Rekurze (přímá i nepřímá) je povolená. Je proto jednoduché modelovat stromové struktury (dokonce i nekonečně vnořené) a jakékoliv jiné grafy.
- Není nutné modelovat podmínku pro zastavení rekurze, protože CSDDM je pouze model. Databázové systémy implementující CSDDM by však měly interně ukládat zarážku jako podmínku pro zastavení do reference (a to do té doby, než je reference naplněna skutečnou referencí).
- Databáze implementující CSDDM musí podporovat minimálně 1000 rekurzívních zanoření.

4. ***perm***

Význam: oprávnění pro provádění operací nad daným záznamem.
Povolené hodnoty pro kolekce: *no_add_rem* , *add* , *rem* , *add_rem* (*no_add_rem* znamená „ani *add* ani *rem* “, *add_rem* znamená „ *add* i *rem* “)
Povolené hodnoty pro atributy: *no_rw* , *read* , *write* (*no_rw* znamená „ani *read* ani *write* “, *write* zároveň znamená i *read*)
Omezení:

- Povolené hodnoty jsou vzájemně výlučné.



Obrázek 12: Životní cyklus kolekce a atributu

5. uniq

Význam: jednoznačnost atributu založená na kombinaci role úkolu a procesní instance.

Povolené hodnoty:

- *none* – data atributů jsou stejná pro všechny instance procesu a pro všechny role mající vhodná oprávnění (např. běžná data v databázích).
- *yes_procs_no_roles* – data atributů se liší mezi instancemi téhož procesu, avšak zůstávají stejné pro všechny role (např. náhodné ID vygenerované pro danou instanci procesu - všechny role musí vidět stejné ID).
- *no_procs_yes_roles* – data atributů se liší mezi rolemi, avšak zůstávají stejná pro všechny procesní instance.
- *every* – data atributů se liší mezi rolemi a instancemi procesů, tj. jedná se o data pro jednu plaveckou dráhu (např. dočasné úložiště pro výsledky uživatelského vstupu).

6. constr

Význam: množina omezení obsahu dat uložených v tomto atributu.

Povolené hodnoty:

- *none* – běžná data s variabilní délkou
- *string_utf8* , *string_ascii_printable* , *string_password* , *string_...* – řetězec s variabilní délkou
- *bool* , *int32* , *int64* , *int_big* , *float* 64 , *float* 128 , *float_big* ,

… – číslo (celé či s plovoucí řádovou čárkou) s danou bitovou délkou v binární reprezentaci
- *time* – časové razítko podporující čas začínající alespoň v roce 2000
- *PDF* , *CSV* , *JSON* , *XML* , *HTML* , *MP*3 , *JPG* , *PNG* , … – specifické druhy datových celků (v názvu omezení se nesmí objevit verze, pokud se nejedná o nedílnou součást názvu – např. HTML4 či HTML5 nejsou validní názvy)
- *app_some_name* , *app_...* – data soběstačné aplikace, která mají být na straně serveru spuštěná v sandboxu namísto provedení běžné operace čtení (výstup aplikace je zaslán jako výsledek operace čtení)
  - Aplikace může poskytovat i vizuální interaktivní rozhraní - např. servlet s klientskou HTML5 WebGL hrou zabírající polovinu zobrazovací plochy).
- *eval* – seznam DDL[3] instrukcí spuštěných při zápisu.
  - Spuštění garantuje změny databázového schématu v případě úspěchu a neovlivnění konzistence dat, a to ani v případě neúspěchu (COW[4] sémantika).
  - Užitečné např. v případě změny množiny omezení pro jeden atribut z *none* na HTML (typová kontrola je spuštěná paralelně na všechna data korespondující s atributem a může selhat).
  - Zaručuje konzistenci DB schématu (především ve spojitosti s ***consis***) po celou dobu evaluace.
  - Podporuje spuštění „na sucho“ (dry run) – např. pomocí speciální instrukce.
  - DDL pro přechod by mohla být seznamem po sobě následujících událostí spuštěných uživatelským vstupem při modelování pomocí CASE nástroje.
  - Uchová historii odstraněných záznamů (včetně jejich vlastní historie) ve schématu tak, aby jako výchozí chování byla data v databázi pouze přesunuta na oddělené „nedostupné“ místo (což může být zcela jiná databáze jako např. Amazon Glacier).

Omezení:
- Mělo by být jednoduché přidat novou množinu omezení.
- Každá množina omezení je definována jako:

3 Data Definition Language
4 Copy On Write

- Unikátní název dle vhodné konvence (viz. povolené hodnoty).
- Verze dle Semantic Versioning [29] s rozšířením pro zpětnou kompatibilitu[5].
- Popis jak vizuálně interpretovat obsaženou hodnotu (slouží i jako nápověda pro automatickou konstrukci uživatelské rozhraní).
  - Obsahuje i informaci o množství dat, která mají být zobrazená najednou (nerozdělená), protože např. řetězec se jménem bude výrazně menší než řetězec celé knihy a např. obrázek či video bude nutné zobrazit celé.
    - Povolené hodnoty: *none* , *small* , *medium* , *large* , *full*
- Výchozí hodnota.
- Formální turingovsky kompletní kód (doporučený je čitelný existující programovací jazyk) popisující kontrolu dat při vstupu do, popř. výstupu z databáze.
  - Tento kód využívá platformně nezávislý proudové rozhraní (inspirace rourou z POSIX).
  - V případě, že je kód napsaný v reálném programovacím jazyce (a je v souladu se specifikací daného jazyka – včetně např. práce s vlákny), pak by měl být přímo spustitelný na systému, který tento jazyk podporuje (tzn. má ho např. nainstalovaný).

7. init

Význam: výchozí hodnota.
Povolené hodnoty: jakákoliv data, která jsou v souladu s vybranou množinou omezení.
Omezení:

- Hodnota je konstantní (např. číslo) a nebo dynamická (např. aktuální datum).
- Hodnotou může být i formální turingovsky kompletní kód (doporučený je čitelný existující programovací jazyk) vracející hodnotu vyhovující vybrané množině omezení (definice algoritmu musí být formální, avšak nemusí se jednat o plně funkční program).

8. sync

5 Support for backporting (https://github.com/mojombo/semver/issues/163).

Význam: synchronizace dat popsaných záznamem (především referencemi).
Povolené hodnoty:

- *once* – hluboká kopie nově přidané hodnoty, avšak pouze jednou bez jakékoliv pozdější synchronizace
- *copy* – hluboká kopie, plně synchronizovaná s referencovanou položkou
- *cache* – hluboká kopie, plně synchronizovaná s referencovanou položkou avšak pouze někdy (jak často a za jakých okolností je implementačně závislé, avšak implementace musí zajistit, aby aktuálnost kopie odpovídala očekávání koncového uživatele)
- *hardlink* – jednoduchá implementační reference přímo na data v referencovaném záznamu

Omezení:

- *hardlink* je jediná povolená hodnota pro záznamy, které nejsou referencemi.

9. consis

Význam: úroveň konzistence při paralelním procesním přístupu (předpokládá se transakční distribuovaný systém).
Povolené hodnoty:

- *strict* – operace čtení a zápisu čekají na doběhnutí aktuálně běžících operací čtení a zápisu a až poté jsou provedeny (tzn. nelze použít Paxos [30] ani podobné algoritmy)
- *linear* – jako *strict*, avšak pouze s linearizabilní konzistencí [31] (např. Paxos s linearizabilní konzistencí jako např. v Cassandra [32])
- *write* – operace zápisu čeká na dokončení operací čtení nebo zápisu a až poté provede zápis (tzn. je povoleno číst kdykoliv)
- *write_lin* – jako *write*, avšak postačuje linearizabilní konzistence
- *read* – operace čtení čeká na dokončení operací zápisu a až poté provede čtení (tzn. je povoleno zapisovat kdykoliv)
- *read_lin* – jako *read*, avšak postačuje linearizabilní konzistence
- *CRDT* (Conflict-free Replicated Data Type) – podobné *linear*, avšak využívá distribuované vysoce dostupné instance dat garantující synchronizaci mezi uzly, avšak ne okamžitě – např. Riak [33] je podporuje (vhodné pro chat, mapy pro navigaci, atd.)

Omezení:

- Všechna data účastnící se transakce jsou považována dohromady jako jedna atomická hodnota.
- V případě kolekce určuje ***consis*** minimální úroveň konzistence případné transakce zahrnující tuto kolekci (tzn. výchozí hodnotou je nejvyšší úroveň konzistence atributů této kolekce a atributů jejích podkolekcí rekurzivně).
- Reference na kolekci či atribut má stejnou úroveň konzistence jako referencovaná kolekce či referencovaný atribut.
- Tyto podmínky znamenají, že databáze implementující CSDDM musí uchovávat přesná časová razítka pro každý záznam (časové razítko musí být dostatečně konzistentní napříč sítí všech databázových uzlů potenciálně vzdálených tisíce km od sebe) – např. jako Google Bigtable [34].

Následující vlastnosti nejsou součástí datového modelu, avšak jsou doporučené pro jeho praktické využití (zejména pro automatickou tvorbu grafických uživatelských rozhraní či optimalizace indexů a kešování). Do vizuálního a strukturního modelu by se promítnuly jako další sloupce.

10. quant

Význam: odhadované množství položek tohoto záznamu v reálně nasazeném systému.
Povolené hodnoty: celá kladná čísla s nulou.
Omezení:

- 0 označuje neznámé množství.

11. impo

Význam: očekávaná důležitost záznamu v kontextu všech ostatních záznamů použitých v daném úkolu.
Povolené hodnoty: *very_low* , *low* , *medium* , *high* , *very_heigh*

Vybrané případy osvědčeného použití.

- Je doporučené řádně promyslet strategii tvorby identifikátorů s ohledem na budoucnost (např. jak bychom volili prefixy a skládání řetězců pro systém o velikosti SAP).
- Vybrané situace pro volbu vlastnosti uniq:
  - *none* – běžná data

    - *yes_procs_no_roles* – náhodně generované ID pro instanci procesu (všechny role potřebují vidět stejné ID)
    - *no_procs_yes_roles* – osobní nastavení jazyka a překladů
    - *every* – dočasné úložiště pro uživatelské vstupy
- Není vhodné modelovat transformace (spojování, dělení, aplikace dalších operací nad daty) jako oddělené záznamy, protože tyto sémanticky nejsou součástí modelu.
- Např. pro překlady by měla být v modelu vybrána pouze jedna řeč a reprezentující výchozí obsah (no_procs_yes_roles) a samstatné překlady by měly být uloženy v oddělené globální kolekci a měly by být aplikované (přepsat výchozí obsah) až při běhu systému (např. pomocí skriptovacího úkolu).

Za hlavní výhodu modelu CSDDM můžeme považovat jeho celistvost a komplexní pohled na celý systém. Zároveň se však jedná i o hlavní nevýhodu, protože se model může zdát nepřehledný (tento nedostatek by měl být výrazně zmírněn využitím jednoduchých grafických symbolů namísto písmen pro atributy, které jsou dostatečně intuitivní – např. ***perm***).

Další vlastností modelu, kterou lze za jistých okolností považovat za nevýhodu je podpora pouze pro implicitní temporální vlastnosti, což je způsobeno podporu paralelního přístupu a zajišťování konzistencí na úrovni jednotlivých záznamů, kde je téměř nevyhnutelné uchovávat u každého atributu časová razítka - obdobně, jak to implementuje např. Datomic [35]). CSDDM se však k temporální vlastnostem staví natolik vysokoúrovňově (např. absencí dotazovacího jazyka), že hlubší podpora v modelu není zapotřebí.

4 Kombinované modely

Při vytváření fungujících programů bylo v minulosti používáno nejprve programovacích jazyků a pamětí, které podporovaly obdobné uspořádání dat (ačkoliv měly různou rychlost), a tak nebylo nutné se zabývat samostatnou strukturou dat. Kapacita pomalejších pamětí však začala rychle růst, a proto se objevil požadavek na efektivní uspořádání dat na pomalých zařízeních. Tato problematika se ukázala natolik komplexní a velice žádaná, že vzniklo samostatné odvětví zabývající se databázemi, uspořádáním a práce s takto

uloženými daty.

Paralelně, avšak pozvolněji se začal objevovat požadavek zefektivňování především logistiky a později celých výrobních procesů za pomocí IT prostředků. Toto odvětví se především zabývalo minimalizací časových prodlev, identifikací a nasazováním paralelizace zpracování úloh a eliminací úzkých hrdel (např. opakování krátkého úkonu mnohokrát za sebou, a to zbytečně).

Tyto dva směry se nepříliš často setkávaly a vyvíjely vlastní modely odděleně. S příchodem masifikace informačních systémů se však tato situace změnila a jsou hledány metody jak tyto dva směry spojit. Ukazuje se, že propojení těchto dvou oddělených ekosystémů není zcela triviální. A to již jenom z důvodu, že např. vývoj modelů dat se výrazně zpomalil a nestihl pokrýt moderní požadavky.

Např. relační datový model nelze ve své čisté podobě paušálně použít, protože nepodporuje uspořádání (na jakékoliv seskupení dat je nahlíženo jako na množinu těchto dat), nepodporuje hierarchické uspořádání dat. Zároveň jeho implementace je natolik komplexní, že neumožňuje zodpovídání rychlých agregovaných dotazů ani např. dostatečně granulární nastavení přístupových práv.

Z těchto a dalších důvodů se objevuje mnoho nových tzv. NoSQL databází, které se snaží uniknout relačnímu uspořádání dat a vyzkoušet různé další směry. Více v kapitole o existujících modelech.

Možná využití kombinovaných modelů zahrnují modelování jakéhokoliv informačního systému, automatizaci, integraci, orchestraci (např. SOA), CMS, DMS, ERP, CRM, a mnohé další systémy, které hodně inteagují buď s uživateli či s ostatními systémy schovanými za nějakými aplikačními rozhraními.

Hlavní výhody kombinovaného modelování:

- Plná nezávislost na platformě či GUI.
- Okamžitě dostupná aplikace (za předpokladu použití nástroje, který převede kombinovaný model na reálnou aplikaci [2]).

- Není nutná verifikace (zbývá pouze validace).
- Odpadá velká část testování (funkcionální, unit, black box) – zůstává však uživatelské testování a integrační/deploy tostování.

Hlavní nevýhody kombinovaného modelování informačního systému:

- GUI není tvořeno člověkem, a je tak neohrabané.
- Na analytika jsou kladeny výrazně větší nároky, protože není omezen technologiemi a musí tedy rozhodnout v mnohem více případech (včetně mnoha detailů).
- Databáze podporující všechny možnosti modelu jsou velice náročné na implementaci (momentálně žádná taková neexistuje, ale s trochou snahy bude velice dobře použitelná např. Datomic).

4.1 Abstraktní požadavky a vlastnosti

Kombinované modely mají rozšířené cíle než pouhé sloučení modelů pro popis dat a popis procesů. Velká část z nich vychází z konzistence, která pouhým sloučením nebude zachována a další vychází z moderních požadavků ubírajících se směrem MDE (Model-driven Engineering). Definovány byly tedy následující schopnosti vyjádření, které jsou sledovány při evaluaci a návrhu kombinovaných modelů.

1. schopnost zachytit jakákoliv data (jak strukturovaná, tak nestrukturovaná, ale především částečně strukturovaná)
2. schopnost zachytit činnosti nad daty a zároveň činnosti zcela bez účasti dat
3. schopnost zachytit závislost i nezávislost jedné činnosti na jiné
4. schopnost zachytit, že na dvou či více různých místech v modelu pracuji s týmiž daty
5. schopnost zachytit pořadí činností (data flow diagramy ho nemají), BPMN ano
6. plně konzistentní zachycení práce s jakoukoliv datovou položkou na

jakémkoliv místě v celém modelu (např. BPMN 2.0 je nekonzistentní, protože kupř. podmínky v bránách využívají booleovskou datovou informaci, která je zcela nezávislá od ostatních datových položek jako např. Data Object)

7. schopnost konzistentně zachytit součinnost uživatelských rolí činností a přístupových práv k jednotlivým atomickým datovým položkám
8. schopnost abstrahovat (nahradit něčím jednodušším) opakující se, často nízkoúrovňové vzory pro zachování přehlednosti a udržitelnosti
9. schopnost zachytit další informace pro využití v MDE [12]

4.2 Historické modely a jejich souvislosti

V historii byl kladen důraz na komplexní modely především u producentů modelovacích nástrojů na rozdíl od výzkumníků. Důvody mj. zahrnovaly komplexitu takové činnosti, velký překryv s programovacími jazyky (zejména těmi implementujícími homoikonicitu) a malém zájmu ze strany komerční.

Z tohoto důvodu se též mnohé modelovací nástroje uchylovaly k ad-hoc řešením nevyzkoumaných oblastí a vznikaly tak nepopsané a aplikačně-specifické modely různých vlastností. Dále budou názvy modelů a názvy modelovacích nástrojů používány ve stejném kontextu.

V dnešní době je však stále kladen větší a větší důraz na automatizaci, a to se zájmem jak na straně komerční, tak na straně výzkumu.

Vybrané modely a modelovací nástroje v historickém sledu jsou zachyceny v následující tabulce. Z důvodu přehlednosti nebyly do výběru zařazeny modely a nástroje, které jsou či byly používány, avšak nejsou rozšířené a zároveň neodpovídají výše definovaným abstraktním požadavkům. Nezařazenými modely a nástroji jsou WCSI, WSFL, XLANG, WSDL, IFML, PNML (Woped tool).

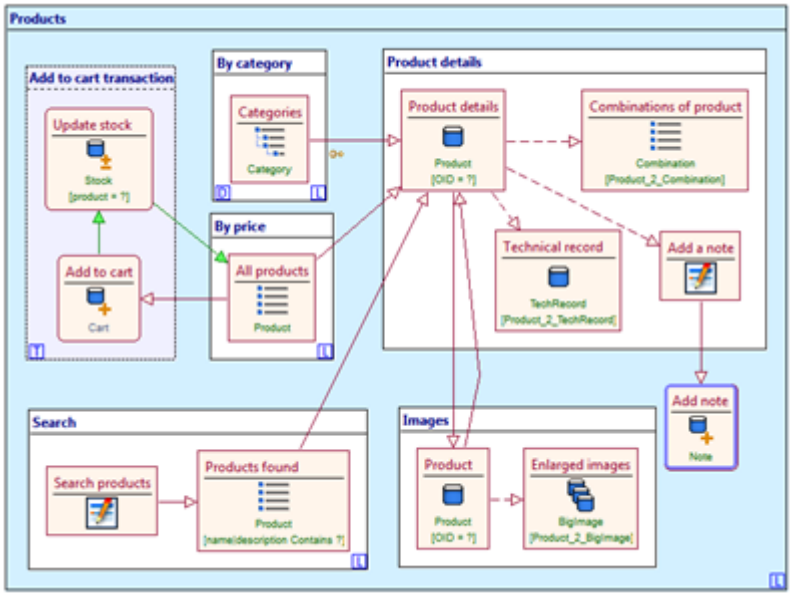
Název modelu či nástroje	Modelování procesů	Modelování dat	Soulad s abstr. pož.	Přibližný rok zveřejnění
DFD [8] a ERD [9]	ano	ano	malý	1970/1976
UML [10] (diagram aktivit a diagram tříd)	ano	ano	malý	1996
SEAM (State-Entity-Activity-Model) [1]	ano	ano	střední	2001
WebML (hypertextový model a strukturální model) [6]	ano	ano	střední	2002
YAWL [7]	ano	ano	střední	2004
BPMN [13]	ano	ne[6]	střední	2004
JBPM3	ano	ano	střední	2010
BORM (OpenCASE) [11]	ano	ano	malý	2012
XPM	ano	ano	úplný	2016

Tabulka 3: Historický sled kombinovaných modelů

Za zmínku stojí metodologie SSADM (Structured systems analysis and design method) [4], kterou lze též použít pro modelování komplexních systémů. SSADM je metodologie zaměřená na obchodní kontrakty a lze ji tedy chápat jako modelování souhrnu pravidel (podobnost s Business Rules [5]), rozpadu na strukturované celky, definic rozhraní, definic procesů a slovních strukturovaných popisů dat. Tato velice propracovaná metodologie je však vysoce komplexní a velice vzdálená technické a implementační stránce systému. Z tohoto důvodu byla nahrazena novějšími metodami zmíněnými v tabulce.

6 Data není možné modelovat, avšak notace obsahuje podporu pro zachycení informace, že se s nějakými důležitými datovými objekty pracuje a kde tato data přibližně nalezneme.

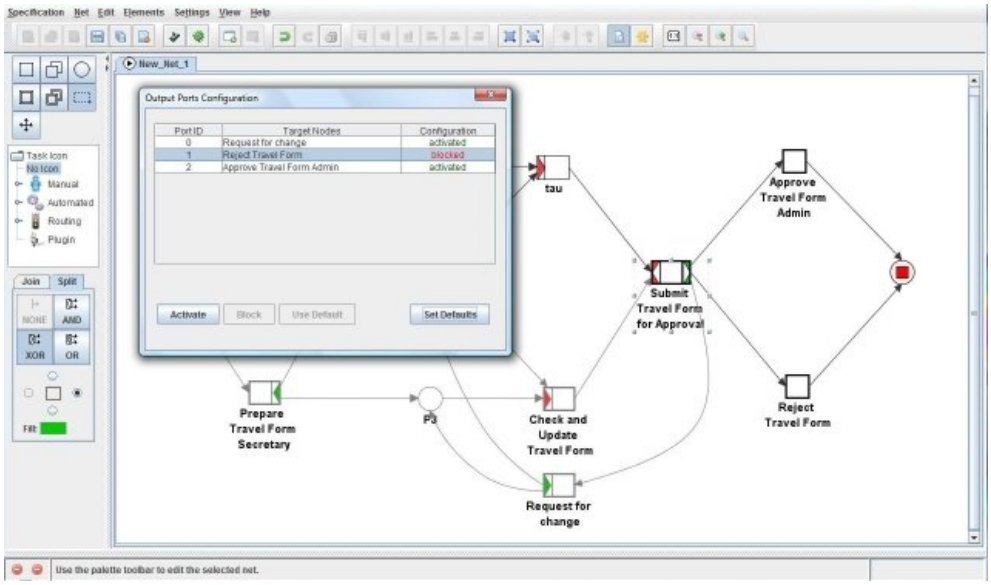
Každá z uvedených modelovacích techonologií pojímá modelování trochu z jiného pohledu. Např. WebML se zaměřuje na konstrukci uživatelského rozhraní výhradně na základě hypertextu a je tedy tímto striktně omezená. Tomu je poté podřízen i způsob modelování procesů (pomocí hypertextových odkazů, vnořování částí hypertextu atd.) a dat (strukturální model, což je jednoduché rozhraní připravené na použití se stávajícími modelovacími technikami – ERD, ODMG objektový model, UML diagramy tříd).



Obrázek 13: WebML strukturální a hypertextový model [16]

YAWL vychází z Petriho sítí [15] a konzistentním způsobem umožňuje pracovat s formuláři jako abstrakcí dat. Model je vystavěný na dlouholetém výzkumu modelování procesů Petriho sítěmi na Nizozemských univerzitách a všechny klíčové části YAWL jsou formálně ověřeny. Vzhledem k silné návaznosti na Petriho sítě jsou YAWL modely jednoduše verifikovatelné a simulovatelné, což je přímo podporováno i referenčním nástrojem pro tvorbu YAWL modelů. Tento model působí nekomplikovaně a přímočaře (princip modelování je poměrně nízkoúrovňový), avšak má ve skutečnosti omezené použití pouze pro webové informační systémy s uživatelskými

vstupy a výstupy skrze formuláře, protože chybí jakákoliv podpora interakce s externími systémy (včetně zdrojů dat), univerzální podpora automatických aktivit a též veškerá omezení vyplývající z modelování dat dle hypertextových formulářů.



Obrázek 14: YAWL modelování procesů

Abstraktní požadavaky na kombinované modely cíleně reflektují dnešní potřebu přiblížit se výrazně implementaci a uživatelskému rozhraní. Např. UML je často vytýkáno [14], že není využíváno v praxi, protože nemá dostatečnou návaznost na pozdější implementaci a výrazně podporuje způsob vývoje, kdy se předem pracně nadefinuje systém a tento se až poté implementuje, přičemž pozdější změny v modelu se velice špatně reflektují v implementaci. A to proto, že implementátoři obdrželi zejména model a nikoliv „vyšší myšlenku“ (jejíž výrazně zjednodušenou variantu model zachycuje), a tedy nemohli předvídat změny na vyšší úrovni než představuje model. V případě UML jsou modely poměrně nízkoúrovňové (až na úroveň metod či jednotlivých malých objektů), což pouze přispívá ke zkostnatělosti tohoto způsobu vývoje.

Výjimkou je nový model XPM, který je navržen tak, aby holisticky zachycoval vysoce abstraktní pohled na systém a nestrukturovaná data (pro

případy, kdy tato rozhodnutí nejsou klíčová a lze je ponechat na implementátorovi či na osobě využívající systém po jeho realizaci), avšak zároveň umožnil velice hluboce a formálně nadefinovat vysoce důležité strukturované detaily a vlastnosti systému.

Z tabulky je též zřejmé, že např. ještě v roce 2001 nebylo možné modelovat celé informační systémy na jednom místě nějak konzistentně. Tento problém tedy mj. vyžadoval velice zkušeného analytika, který byl schopen velké oddělené modely dat a procesů v hlavě konzistentně propojit, vyvíjet a udržovat. Riziko chyby tedy roste exponenciálně s přidáním každé nové položky do jednoho z datového či procesního modelu, protože potenciální propojení mezi položkami těchto modelů je M:N.

4.3 Nový kombinovaný model XPM 1.0

Novým počinem na poli kombinovaných modelů je model XPM 1.0, který se snaží konzistentně zachytit všechny abstraktní požadavky a vlastnosti a to zároveň při zachování jak přehlednosti, tak potřebné formality.

XPM vychází z procesního modelování *BPMN 2.0*, do kterého se snaží začlenit podporu pro ucelený, univerzální, avšak přesný způsob modelování dat. To vše s cílem co nejméně upravovat *BPMN 2.0* (ani notaci, ani sémantiku).

Specifikace *BPMN 2.0* nabízí zcela základní nástroje na zaznamenání informace, že se v procesu pohybují data, kde jsou hlavní data uložena a jaké jsou asociace mezi úložnými místy a místy použití těchto dat. Tyto nástroje však neposkytují žádné formální prostředky pro dostatečně podrobný popis samotných dat, nikoliv pouze hrubé rozhraní míst interakce s procesními prvky.

V *BPMN 2.0* tedy nalezneme následující pojmy.

- Data Store
- Data Object
- Data Association

- Data Input
- Data Output

Vzhledem k výše uvedeným limitacím např. na umístění *Data Store* vždy do bazénu, kde jsou data používána nelze dostatečně obstraktně modelovat dnes běžné případy externích úložišť. *BPMN 2.0* v takových případech vyžaduje vytvoření zástupce, proxy, který bude vytvářet rozhraní mezi externím úložištěm a lokálním bazénem. Ani toto řešení však zcela nevyhovuje, protože dnešní business aplikace, na které *BPMN 2.0* cílí, se provozují ve virtualizovaných prostředích s plovoucím umístěním datových úložišť a měnícími se rozhraními.

Právě z těchto důvodů je pro potřeby *XPM* rozšířena sémantika prvku *Data Object*, který má dle *BPMN 2.0* následující dva základní způsoby použití.

1. Nesměrová informace, že daný úkol (*BPMN 2.0 Task*) pracuje s či jinak využívá daný logický datový celek.
2. Směrová informace, že daný logický datový celek je přenášen z jednoha místa na druhé (nelze rozlišit kopii a přesunutí).

Rozšíření sémantiky je aplikováno na první případ – nesměrovou informaci – kdy prohlásíme, že existuje speciální varianta *Data Object*, který popisuje samotná data do takové míry, že je možné tento logický popis převést do fyzického datového uspořádání v libovolném databázovém systému podporujícím použitou podmnožinu funkcionality v logickém popisu dat. Každý *BPMN 2.0 Task* může mít asociaci maximálně s jedním tímto speciálním *Data Object*. Tento speciální *Data Object* nazýváme *XPM Data Object* (dále jako *XDO*).

Tato definice nevylučuje použití běžných *Data Object* u *BPMN 2.0 Task* i v případě, že je již použitý jeden *XDO*. Takovéto použití se však nedoporučuje, protože může být matoucí a též i redundantní.

XDO nám dovoluje chápat jednotlivé úkoly (*BPMN 2.0 Task*) jako samostatné celky pracující s určitou fixní podmnožinou dat. Jednotlivé *BPMN 2.0 Task* mohou navzájem jednotlivé části fixních podmnožin referencovat (ať již jako odkaz či automaticky synchronizovaná kopie), a to

i napříč různými procesy a bazény. Specifikace dat v *XDO* plně podporuje vytváření kolekcí (zhruba odpovídajících tabulkám v relačních databázových modelech), čímž je mj. zajištěno, že není potřeba více než jednoho *XDO* na jeden *BPMN 2.0 Task*.

Celkové chápání *Data Object* se tedy posunulo z názvu logického celku vzájemně souvisejících dat k přesnému popisu všech dat, se kterými daný *BPMN 2.0 Task* pracuje.

Data popsaná ve všech *XDO* se v rámci jednoho procesu a rozbalených podprocesů (*BPMN 2.0 Expanded Subprocess*) považují za globální a jsou tedy referencovatelná bez nutnosti uvádění prefixu s identifikátorem procesu. Toho lze s výhodou využít např. v podmínkách bran (*BPMN 2.0 Gates*), přičemž v případě použití jen jediného *XDO* v danémo procesu je zakázané využívat jakýchkoliv jiných datových zdrojů než *XDO* objekty a kompatibilní konstanty.

V *XPM 1.0* není zahrnuta podpora pro introspekci samotného procesu, protože je to považováno za nebezpečnou a nepříliš užitečnou vlastnost. V případě potřeby ve speciálních aplikacích lze tuto funkcionalitu implementovat programově v závislosti na zvolené technologii.

Dále v *XPM 1.0* nenalezneme žádnou parametrizaci modelovaného systému. Parametrizace by se s výhodou dala využít např. pro přidání dodatečných UI prvků do výsledné aplikace (prkem může být tlačítko pro import/export, centrální vyhledávání, či jakýkoliv jiný prvek, který souvisí s daty, avšak není modelem zachycený a tedy je postradatelný). Tato funkcionalita může být přidána v budoucích verzích *XPM*.

Následují některá nezřejmá pravidla pro zachování konzistence v modelu *XPM 1.0*.

- Všechny použité algebraické a relační operátory ve formálních výrazech v bránách (podmínkách) garantují, že všechny jejich operandy splňují odpovídající *CSDDM 1.0* omezení.
- *XDO* je k *BPMN 2.0 Task* vždy připojený čárou mající na obou koncích šipky.

5 Modelovací nástroje

Modelovací nástroje jsou aplikace, které umožňují vhodným způsobem pracovat s existujícími teoretickými modely a snaží se usnadnit jejich tvorbu a přiblížit se tak k reálné aplikaci těchto modelů ve formě fungujícího systému. K této transformaci z teoretického modelu do realné aplikace jsou bez zpravidla potřebné nemalé prostředky, jejichž prostřednictvím je transformace realizována.

Hlavním přínosem modelovacích nástrojů je vizualizace, nejčastěji ve dvourozměrném eukleidovském prostoru, automatizované hlídání nekonzistencí v modelech a zároveň možnost vhodného propojování modelů do vyšších ucelenějších a popisnějších celků.

Modelovací nástroje propojující více modelů poté nazýváme kombinovanými modelovacími nástroji. V této práci se dále budeme zabývat takovými kombinovanými nástroji, které kombinují datové a procesní modelování a případně další modely s tím související (např. diagram případů užití).

Tyto nástroje zejména zapadají do skupin tzv. CASE (Computer-aided software engineering) nástrojů, jejichž cílem je umožnit nebo přímo zprostředkovat tvorbu software, který má vysokou kvalitu, je bezchybný a dobře udržovatelný. To vše při zachování krátké doby vývoje.

S kombinovanými nástroji se setkáváme nejčastěji ve velkých podnicích, kde je zapotřebí zachytit celistvým způsobem realitu a nikoliv pouze teoretické a vzájemně nesourodé oddělené části reálného fungování provozu či organizace.

5.1 Metoda porovnávání – zkušební případ

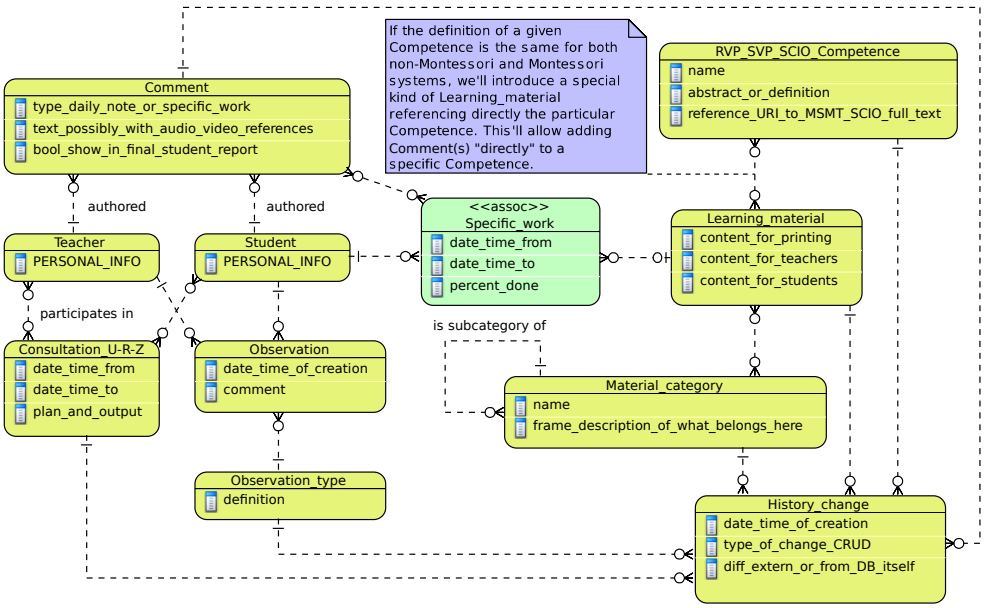
Porovnávání kombinovaných modelovacích nástrojů proběhlo na případu modelování vybraných oblastí informačního systému pro Montessori vzdělávací zařízení. Fungování těchto zařízení se v mnoha ohledech liší např. od zařízení státního vzdělávání, která jsou si velice podobná a vykazují výrazně vyšší stupeň standardizace při práci s daty. Právě kvůli

větší rozmanitosti bylo zvoleno modelování právě tohoto vzdělávacího systému.

Kritéria porovnávání zahrnovala následující.

- Efektivita tvorby modelů (např. kolik času je potřeba na tvorbu jednoduchého modelu).
- Možnosti tvorby modelů (které části nelze vytvořit nebo lze, avšak s obtížemi).
- Interoperabilita (kompatibilita s existujícími standardy, rozšiřitelnost o novou funkcionalitu, apod.).

Na základě požadavků posbíraných od několika nejaktivnějších českých ředitelek, učitelek a učitelů z různých Montessori zařízení z celé České republiky vzniknul následující datový konceptuální model zachycený ER diagramem. Jeho části byly použity jako referenční model pro evaluace možností daných nástrojů.



Obrázek 15: Referenční datový model Montessori IS

5.2 Stávající nástroje

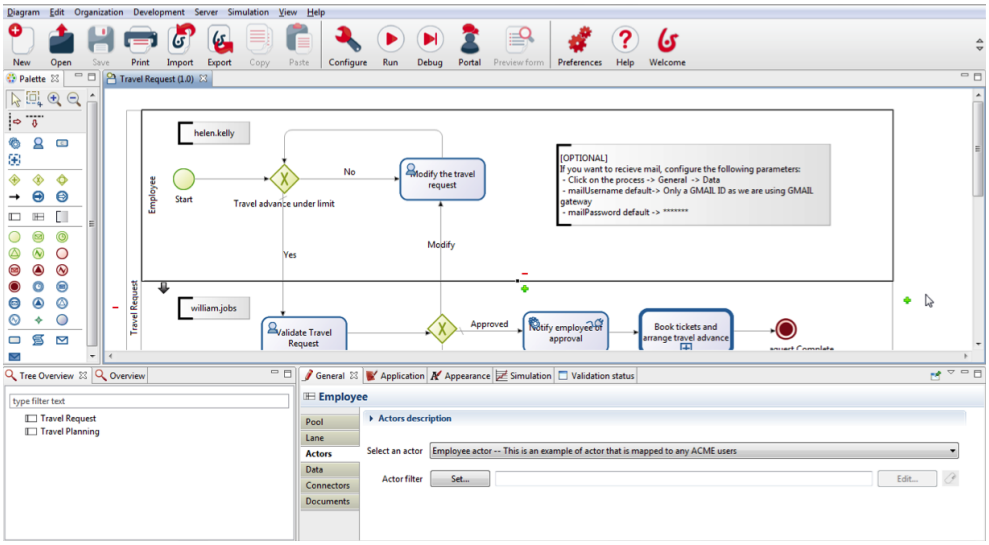
Následující nástroje byly nainstalovány na běžný uživatelský počítač či byly využity profesionálně připravené instalace na erárních počítačích a byl v nich založený nový projekt. Hlavním cílem bylo vyzkoušet jak by bylo možné namodelovat jednotlivé části referenčního datového modelu.

Tyto nástroje byly vybrány díky své dobré dostupnosti a rozšířenosti v oblasti BPM modelování.

1. Bonita Community BPM

   Tento nástroj pocházející z Francie si v posledních letech vydobyl velice dobré postavení díky rychlému a kvalitnímu vývoji a jednoduchému přístupu k modelování procesů, ale i dat, která lze přímo vizualizovat ve webových formulářích. Model se nepodařilo plně zrealizovat, protože modelování probíhá dokumentově orientovaným způsobem, avšak bez referencí a je tedy nutné data pomocí programování (např. skriptů v jazyce Groovy) spravovat a kopírovat (např. pro implementaci asociativních vztahových tabulek).

   - Efektivita tvorby: bez asociativních vazeb a M:N vazeb vysoká, avšak s nimi střední díky nutnosti programování.
   - Možnosti tvorby: široké, ačkoliv stále chybí pokročilé BPMN 2.0 prvky. V žádném případě však možnosti nedosahují tak daleko jako IBM WebSphere či jBPM.
   - Interoperabilita: střední – exportované modely jsou téměř plně odpovídající standardu, avšak import modelů vytvořených v ostatních nástrojích není zcela bezproblémový. Taktéž verzování výstupů, ačkoliv podporované, nevyhovovalo představám o kvalitním verzování software či dokumentů (objevovaly se též chyby ve vynucování verzí).

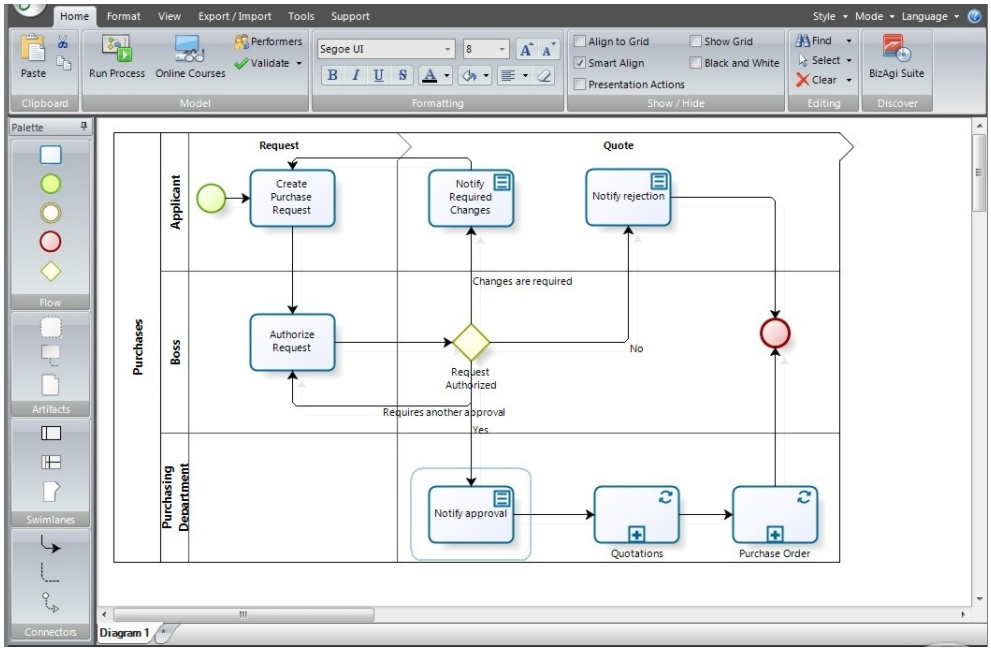


Obrázek 16: Bonita Community BPM [20]

2. BizAgi

   Tento poměrně pokročilý nástroj je zajímavý mj. svou multiplatformností – ke svému běhu vyžaduje buď CLR nebo JVM. Podporuje různé relační databázové backendy s podporou SQL. Ačkoliv se jedná o déle vyvíjený a větší nástroj, jeho možnosti měly obdobné limitace jako Bonita Community BPM, tedy např. nemožnost bez programování definovat asociativní vztahy.

   - Efektivita tvorby: bez asociativních vazeb a M:N vazeb vysoká, avšak s nimi nízká (programování pro BizAgi je náročnější a dále ztížené komplexní navigací v uživatelském prostředí)
   - Možnosti tvorby: velice široké s možností dalšího rozšíření.
   - Interoperabilita: import a export funguje téměř bez obtíží, avšak v případě výskytu obtíží jsou tyto téměř neřešitelné vzhledem ke komplexitě chyb, které Bizagi hlásí. Tento fakt dělá větší projekty neudržovatelnými.



Obrázek 17: BizAgi Process Modeler [21]

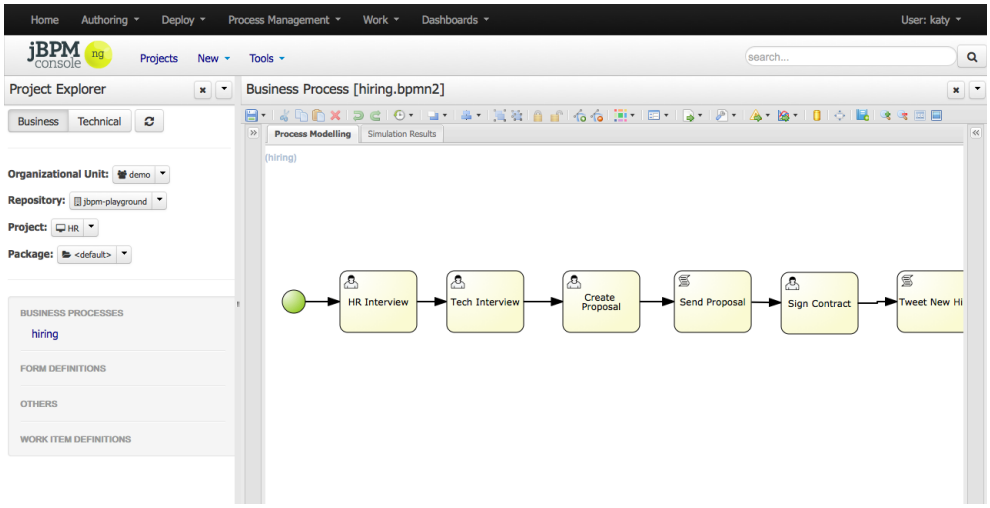
3. jBPM

   Tato platforma je součástí velkého middleware balíku Jboss a snaží se být konkurencí obdobných velkých řešení např. od IBM s tím rozdílem, že všechny součásti jsou plně open-source. jBPM je skutečně první ze dvou nejpokročilejších řešení, které podporuje vše co je potřebné pro úplnou implementaci procesů a dat. Škoda jen, že rozhraní je poměrně neintuitivní a že pro kterýkoliv detail je nutné programovat v Javě minimálně několik řádků kódu (např.).

   - Efektivita tvorby: tvorba procesů je přímočará a rychlá. Tvorba dat je však výrazně komplexnější, protože vyžaduje jak definici datových položek v tabulce (identifikátor, typ, hodnota, databázové uložení, napojení na ostatní) pro každou aktivitu, tak přiřazení k formulářovému výstupu či přímo naprogramování logiky s datovou položkou související. Asociativní vazby, rekurzivní vztahy apod. je též nutné obsluhovat programováním.
   - Možnosti tvorby: veliče široké, avšak zdlouhavě a nepříliš

intuitivně dostupné.

- Interoperabilita: nejlepší ze všech testovaných řešení. Tvůrci mají zájem, aby modely byly přenositelné oběma směry.



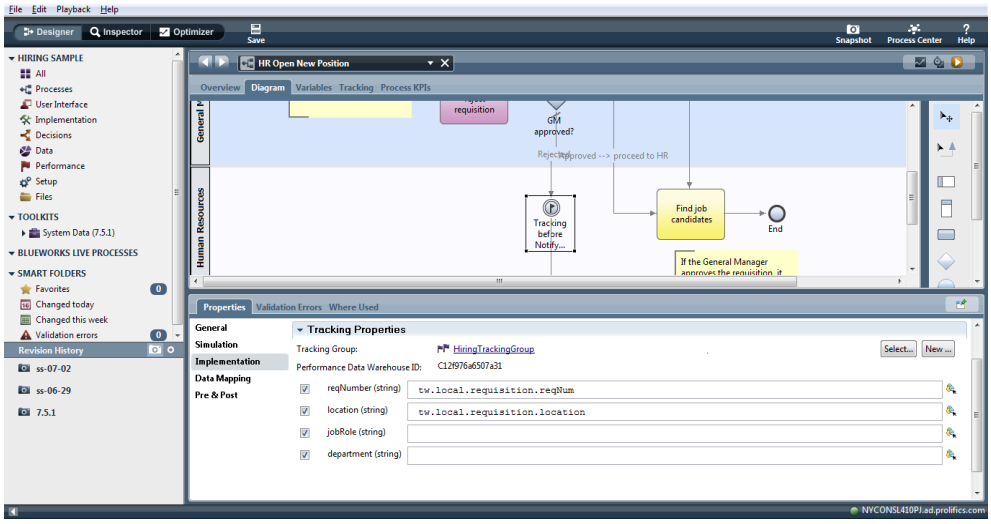
Obrázek 18: jBPM Console NG [22]

4. IBM WebSphere

   Nejdéle vyvíjený nástroj ze všech testovaných a tomu i odpovídají možnosti a zpracování. Pokud je třeba nástroj, který umí téměř vše, pak to je určitě tento. Nevyhneme se však některým nepříjemným problémům. WebSphere implementuje některou funkcionalitu jinak než je ve specifikaci a přidává do modelů vlastní konstrukce. Tímto poměrně zásadním způsobem omezuje výměnu modelů mezi různými nástroji a dále ztěžuje jeho použití těm, kteří znají standard.

   - Efektivita tvorby: srovnatelná s jBPM, a to především po stránce nutnosti využívat kód (Java) pro dosažení konstrukcí složitějších než pouhé struktury atomických datových typů. Stejně jako u jBPM pozorujeme silnou závislost na možnostech platformy Java, což je velice limitující např. pro práci se semi-strukturovanými daty.
   - Možnosti tvorby: velice široké, avšak obtížněji dostupné.

- Interoperabilita: vzhledem k mnohým proprietárním rozšířením špatná. Při pokusu o dodržování standardů exportované soubory přesto neodpovídaly standardu a bylo nutné provést manuální úpravy v XML.



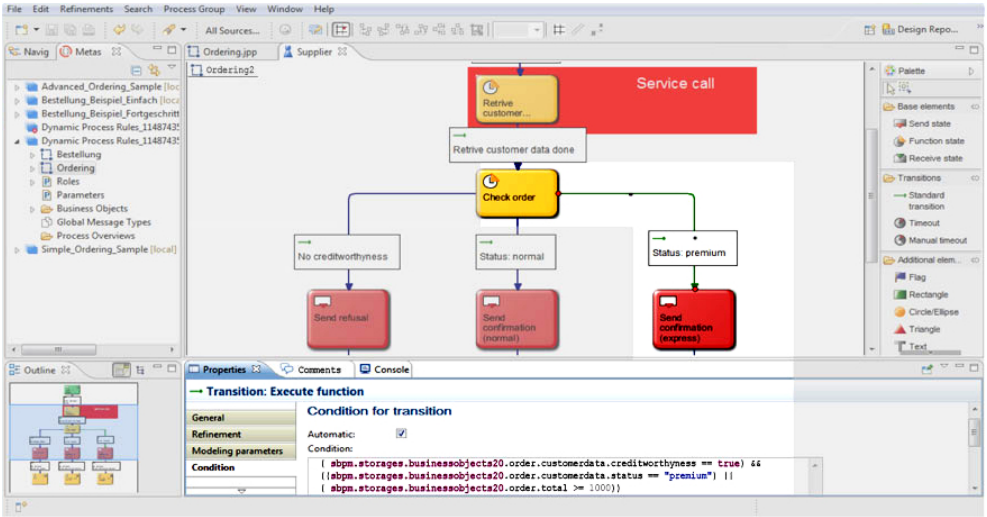
Obrázek 19: IBM WebSphere - Business Process Designer [23]

5. Metasonic

   Tento nástroje zde zařazen především jako ukázka moderního pojetí procesního přístupu k tvorbě funkčně a uživatelsky zaměřených aplikací. Tvůrci pojali procesní modelování dat vysoce pragmaticky a vytvořili novou procesní notaci připomínající zjednodušené BPMN 2.0, přidali jednoduché modelování dat a u toho všeho se snažili, aby modelování mohl provádět i nezkušený člověk, který není analytikem a možná ani ne osobou pracující v IT. Metasonic nazývá toto modelování S-BPM a jedná se o ekvivalent stavového automatu. Metasonic nástroje se snaží vytvářet modely, které jsou okamžitě plně spustitelné na mobilních platformách a které bezproblémově zajišťují základní úkony (přihlásit se, stáhnout seznam, zobrazit, nechat uživatele vybrat položku, změnit ji, uložit změnu, odhlásit se).

Díky tomuto přístupu vytvořili zřejmě nejužitečnější procesně a datově orientovaný nástroj dneška. Bylo potěšením s nástroji pracovat.

- Efektivita tvorby: velice vysoká a nebýt nemožnosti modelovat asociativní vztahy, nebylo by co vytknout.
- Možnosti tvorby: středně velké – nástroje se zaměřují na menší, zpravidla mobilní aplikace a tudíž zde nenalezneme aplikační servery ani podobné záležitosti. Pokročilejší prvky lze doprogramovat, avšak tomuto se snaží nástroj všemožně vyhnout a není to tedy doporučený postu. Tento fakt též zamezil plně implementovat referenční model a jako jediný testovaný nástroj tedy nesplnil předpoklady pro univerzální modelovací nástroj.
- Interoperabilita: špatná – nevyužívá se žádná standardizovaná notace ani pro export ani pro import. Je proto nutné případné modely překreslit. Nástroje jsou však natolik intuitivní a příjemné na práci, že po zkušenostech s předešlými nástroji bylo překreslování spíše zábavou než nutnou prací.



Obrázek 20: Metasonic Suite 5.0 [24]

Mezi dalšími existujícími nástroji, které nebyly testované, avšak začínají si

nacházet cestu k uživatelům nalezneme např. Signavio, online procesní nástroj, který se však rigorózně drží existujících modelů, a proto neposkytuje možnost propojeného modelování procesů a dat. Signavio platforma je však velice nadějná a do budoucna bude zcela jistě konkurencí k výše uvedeným řešením.

Výsledkem porovnání existujících řešení je, že neexistuje známý nástroj, který umožňuje bez větších obtíží (např. bez programování) modelovat procesní práci s daty. Toto zjištění potvrzuje, že procesní pohled na systémy a datový pohled na systémy byly historicky vzájemně oddělené a doposud se nepodařilo je dostatečně úspěšně spojit.

6 Vyvíjený CASE nástroj pro XPM

Pro vytváření nových modelů *CSDDM 1.0* a *XPM 1.0* bylo hojně využíváno znalostí ze zpětné vazby získáváné z testování a počátečního vývoje CASE nástroje, který by měl po dokončení sloužit jako plnohodnotná referenční implementace kombinovaného modelu *XPM 1.0*. Jedná se tedy především o modelovací nástroj pro *BPMN 2.0* s rozšířením sémantiky o *CSDDM 1.0* anotace. Snaha je kladena na zajištění volitelnosti všech novinek, které *XPM 1.0* oproti *BPMN 2.0* nabízí (tedy především celý datový model *CSDDM 1.0*).

Tento CASE nástroj je označen XPMA-CASE, protože jeho výstup by měl být použitelný pro účely projektu XPMA [2]. XPMA je projekt, který z definic procesů a dat ve formě modelů bude generovat parametrizované implementace těchto vstupních modelů. Vygenerované implementace budou pokrývat různé platformy (včetně mobilních) a operační systémy, což je umožněno práve parametrizací. To vše při garantování, že implementovaný systém (ať již na kterékoliv platformě) je přesně ve vztahu 1 : 1 k tomu, co analytik chtěl.

XPMA-CASE by kromě plné podpory *XPM* měl umožňovat i zprostředkování zjednodušených náhledů na modelovaný systém pomocí známých ER diagramů pro namodelovaná data a Use-Case diagramů pro přehled o procesech. Pro účely XPMA-CASE jsou Use-Case diagramy

chápány jako relace procesů a rolí.

Vývoj nástroje XPMA-CASE v rámci této práce byl zaměřený na evaluaci principů propojení datového modelu *CSDDM 1.0* a procesního modelu *BPMN 2.0*. Cílem tedy nebylo vytvořit plnohodnotný modelovací nástroj, nýbrž nastínit, jakým směrem by se vývoj takového nástroje měl ubírat.

Pro vývoj byla zvolena volná vývojová metodika založená na agilních myšlenkách. Základním principem použité metodiky bylo krátkodobé plánování (jeden či dva týdny dopředu), kdy v rámci této krátké doby byla implementována malá funkcionalita aplikace. K vývoji byl využit verzovací systém *Git* a pro plánování jednoduchý seznam položek, které byly dle potřeby prioritizovány či upozaďovány, rozpadány či slučovány. Vývoj se tedy podobal běžným open-source projektům malé velikosti.

Vývoj neprobíhal dle rigorózní metodiky, protože za analýzou, návrhem, plánováním, vývojem a testováním stála jediná osoba a nebylo tedy nutné řešit koordinaci mezi více členy týmu. Koordinace, sdílení myšlenek, efektivní komunikace a potřeba dosáhnout vytyčených výsledků v daném čase jsou hlavní důvody, proče vývojové metodiky existují, a proto v tomto případě nebylo nutné se k nějaké uchýlit.

6.1 Analýza a návrh

Analýza probíhala formou malé rešerše existujících CASE nástrojů pro procesní modelování a dále vizualizačních knihoven pro ER diagramy. Výsledky této rešerše ovlivnily výběr technologie a programovacího jazyka a též přinesly inspiraci pro jednotlivé vlastnosti prvků XPMA-CASE.

Výhodnou skutečností pro analýzu byl fakt, že velká část funkčních požadavků aplikace byla již dostatečně přesně definována ve standardu *BPMN 2.0*, protože XPMA-CASE má sloužit právě převážně k procesnímu modelování. Níže uvedené funkční požadavky tedy nezahrnují specifika *BPMN 2.0*.

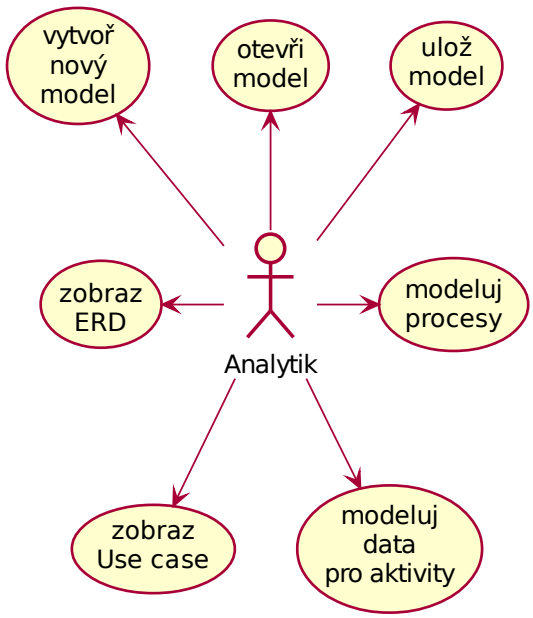
Funkční požadavek	implementováno
Plná podpora modelování kombinovaného modelu *XPM 1.0* (tzn. *BPMN 2.0*, *CSDDM 1.0* a změn viz. kapitola o XPM).	částečně
Podpora exportu a importu *BPMN 2.0* diagramů (v rámci nichž jsou uložené i definice *CSDDM 1.0*).	ano
Podpora exportu diagramů do SVG.	ano
Podpora hlídání pravidel konzistence *XPM 1.0*, *CSDDM 1.0* a *BPMN 2.0* se zpětnou vazbou pro uživatele.	ne
Přidávání záznamů do CSDDM chytře nastaví výchozí hodnoty vlastností (v závislosti na nejběžnějším použití a okolních záznamech).	ne
Interaktivní doplňování existujících identifikátorů (např. při psaní referencí).	ne
Automatické generování a zobrazování ER diagramu se všemi daty z *CSDDM*.	ne
Automatické generování a zobrazování Use-Case diagramu se všemi daty z *BPMN* a *CSDDM* (přiřazení názvů procesů jako jednotlivých případů užití k rolím ze všech plaveckých drah).	ne
Podpora přepínání mezi „business“ pohledem (skrytí *BPMN 2.0 Tasks*, které pouze něco připravují pro ostatní „hlavní“ *BPMN 2.0 Tasks*) a plným pohledem.	ne
Podpora triviálního prohození referencovaného místa s místem na kterém referencuji (míst, na kterých referencuji, může být $0..N$).	ne
Intuitivní a jednoduché přidávání nových množin typových omezení v CSDDM.	ne

Tabulka 4: Funkční požadavky XPMA-CASE

Základní nefunkční požadavky zahrnují následující.

- Multiplatformnost (unixové/linuxové platformy s X11, Microsoft Windows, Mac OS X, Android).
- Uživatelská přívětivost (pohledné UI, odezva UI, logické ovládání, minimalizace pohybů a klikání myší a minimalizace množství stlačovaných ovládacích kláves).
- Open-source pod permisivní licencí (např. MIT).

Hlavním rizikem v tomto projektu je jeho velikost. Samostatná specifikace BPMN 2.0 je natolik velká a komplexní, že pro základní implementaci by bylo odhadem potřeba několik stovek MD. V případě CSDDM 1.0 je situace výrazně lepší (odhadem desítky MD), avšak problém je v závislosti na *BPMN 2.0*, což znemožňuje implementaci dříve než budou připravené potřebné prvky *BPMN 2.0*. Další zvýšení rizika vyplývá ze zaměření na vizuální reprezentaci, což je v SW projektech vždy ta nejdelší a nejúmornější část, protože se detaily musí ladit velice dlouho.



Obrázek 21: XPMA-CASE - diagram případů užití

Z rešerše dostupných nástrojů a technologií se vzhledem k požadavku

multiplatformnosti ukázala jako nejlepší volba webová technologie HTML5, která je dostupná na všech platformách v dostatečně funkční podobě. Vybranou technologií je modelovací a vykreslovací jádro bpmn.io [3]. Toto jádro je naprogramováno pouze za pomocí webových technologií (Javascript a HTML5) a vyvíjeno pod otevřenou licencí MIT s dodatkem, že v odvozených dílech musí být vždy viditelné zelené logo firmy Camunda Services GmBH.

Vzhledem k výběru této technologie byl vybrán Javascript jako hlavní implementační jazyk. Tento jazyk je slabě typovaný a i přes neefektivní návrh poskytuje na běžných uživatelských stanicích dostatečný výkon pro implementaci modelovacího nástroje.

Architektura aplikace sestává z jádra bpmn.io, okolo kterého je postaven „front end“ umožňující načtení a uložení souboru a vytvoření nového diagramu. Front end využívá moderní knihovny Electron [28], díky které je spustitelný na více platformách. Po spuštění aplikace se nejprve spustí prostředí Electron jako server a poté se k němu připojí obalené jádro bpmn.io, pošle mu informaci o položkách menu (které zprostředkuje Electron) a nakonec si vyžádá canvas, do kterého vykresluje diagramy.

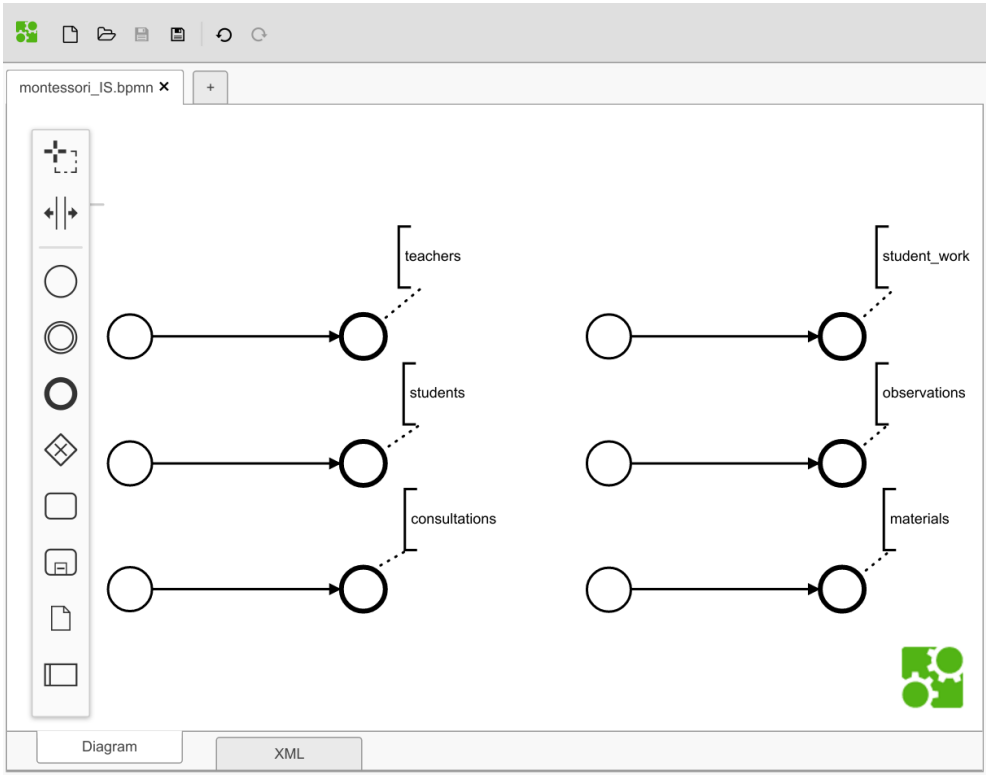
Nejvýznamější třídy využívané v XPMA-CASE jsou následující.

- *Modeler* (jádro bpmn.io – interaktivní práce s diagramy)
- *PropertiesPanel* (panel vpravo, zobrazující vlastnosti jednotlivých grafických prvků)
- *Clipboard* (podpora schránky)
- *PaletteProvider* (výběr modelovacích nástrojů)
- *BpmnRenderer* (vykreslování diagramů)
- *BpmnTreeWalker* (zpracování XML stromu s definicí BPMN procesů)

6.2 Vývoj

Vývoj XPMA-CASE probíhal na platformě Linux X11. V první fázi vývoje, jejíž výstup je popsán v této práci bylo hlavním cílem vytvořit funkční prostředí pro testování a výzkum nových metod modelování dat, které budou jednoduše aplikovatelné na stávající standardy, ale přesto budou poskytovat plnou integraci.

Pro další fáze vývoje XPMA-CASE bylo plánováno rozšíření jeho funkcionality o všechny funkční požadavky a tedy vznik plně funkčního modelovacího nástroje pro kombinovaný model XPM 1.0 s podporou pro dodatečné pohledy ERD a Use-Case.



Obrázek 22: XPMA-CASE – testování vizuálních elementů v rané fázi vývoje

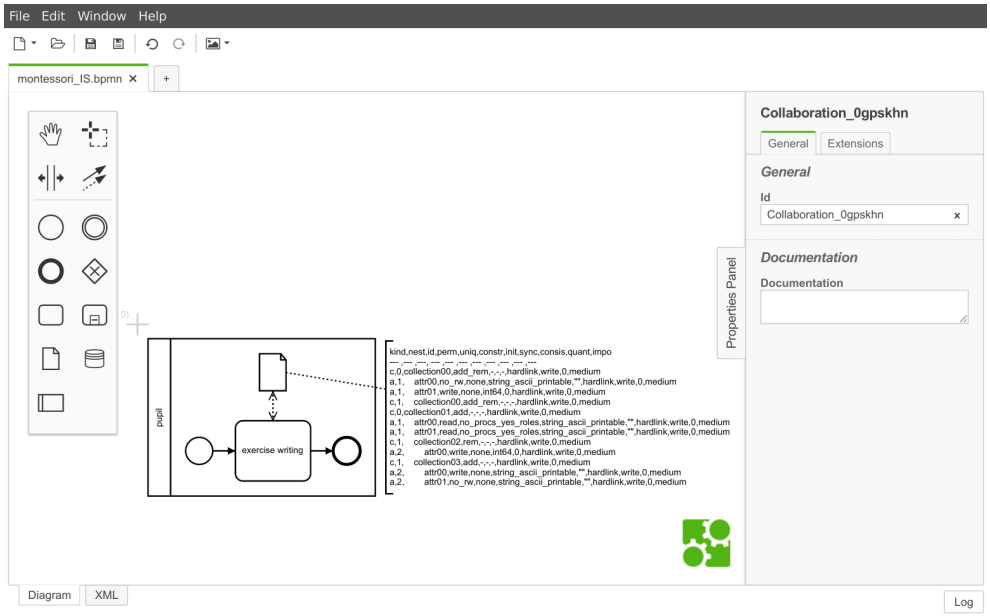
6.3 Testování

Metody testování zahrnovaly zejména funkční testování, black box testování a testování uživatelské přívětivosti. Nebylo nasazeno unit testování, protože se jedná o prototyp a psaní unit testů by v takovém případě zbytečně prodlužovalo vývoj. Pro budoucí vývoj se však s unit testováním počítá.

Nejzajímavější metodou testování z pohledu přínosů bylo testování uživatelské přívětivosti a black box testování na nezaujatých osobách (bratr a spolubydlící). Výsledkem bylo zjištění, že se jedná o příliš technický nástroj, který zprvu není pochopitelný pro osoby nepracující v IT, avšak po krátké chvilce byl i běžný uživatel schopen vymodelovat jednoduché procesy a pochopit co se v nich děje. Pro hlubší pochopení proč nelze dát časovač dovnitř *BPMN 2.0* Task bylo však nutné vysvětlit přesné významy jednotlivých grafických prvků v modelu.

Naneštěstí testování *CSDDM 1.0* nemělo smysl, protože nebyla ještě implementována vizuální podoba a uživatel je tedy konfrontován s nevzhlednou ručně editovatelnou CSV tabulkou.

Celkový dojem však obě testující osoby měly dobrý a rády by se účastnily dalšího testování po naimplementování chybějící funkcionality.



Obrázek 23: XPMA-CASE - XDO s daty v CSV

6.4 Nasazení aplikace

Do této fáze reálného nasazení XPMA-CASE nedospěl, a proto lze pouze zmínit, že v průběhu vývoje a testování nebyly nalezeny žádné zádrhely, které by znemožňovaly nasazení XPMA-CASE na jakoukoliv platformu dle původních požadavků.

7 Závěr

Hlavním přínosem této práce je nový datový model *CSDDM 1.0*, který je navržený pro použití spolu s procesním modelováním systémů a snaží se o zachycení systému dostatečně komplexním, avšak stále přehledným způsobem, aby při implementaci bylo přesně jasné co se od práce s daty očekává.

Pro tvorbu *CSDDM 1.0* bylo nutné vytvořit i demonstraci způsobu napojení na procesní modelování, a tak byl vytvořen kombinovaný model *XPM 1.0*, který demonstruje úzkou spolupráci *CSDDM 1.0* s rozšířeným procesním

modelem *BPMN 2.0*. V kombinovaném modelu *XPM 1.0* byla poupravena sémantika *Data Object* a přidán speciální typ *XPM Data Object (XDO)* zprostředkovávající prostor pro specifikaci dat pomocí *CSDDM 1.0* a dále bylo zajištěno, že všechny výrazy v bránách pro rozštěpení procesů využívají právě těchto dat spolu s konstantami kompatibilními s příslušnými operátory či neformálních slovních popisů (to vše namísto ad-hoc proměnných, které lze dle BPMN 2.0 v procesech použít).

K účelu tvorby CSDDM 1.0 byly evaluovány existující metody modelování dat, procesů a kombinovaných modelů spolu s existujícími nástroji. Na základě poznatků byl model CSDDM 1.0 připraven tak, aby byl nezávislý na technologii, na použitém procesním modelu a umožňoval zachytit systém holistickým způsobem.

Evaluace CSDDM 1.0 probíhala v rámci započaté implementace CASE nástroje XPMA-CASE. Započetí vývoje tohoto nástroje výrazným způsobem přispělo k některým rozhodnutím potřebným k dosažení cílů jako je nezávislost na technologii (ať již databázi, procesním modelu či programovacím jazyku).

Neočekávaným přínosem této práce jsou jednotlivé rešerše provedené za účelem vyhodnocení základních možností pro vlastnosti jednotlivých záznamů v *CSDDM 1.0*. Např. konzistence je parametr, který bývá většinou zcela skrytý a přitom je naprosto zásadní pro určení spolehlivosti systému. CSDDM 1.0 odkrývá tato zákoutí a nutí analytika diktovat podmínky tak, aby byly implementovatelné, což však často vede k polevení a ihned z modelu je poté zřejmé, že s daným datovým záznamem není zcela spolehlivě zacházeno.

CSDDM 1.0 však nemá kompletně nadefinovanou sadu pravidel, dle kterých lze jednotlivé vlastnosti záznamů kombinovat. Vzhledem k vysoké komplexitě a velkému stavovému prostoru možných kombinací bude nutné model buď formálně verifikovat či alespoň empiricky doplnit další potřebná pravidla do sady.

XPM 1.0 taktéž plně nevyhovuje požadavkům analytika. Mimo to, že s sebou přináší všechny nedostatky *BPMN 2.0* (např. nemožnost zabránit

míchání aktivit v případě více nezávislých pracovníků nasazených na stejný proces, avšak v nezávislých instancích), je též zbytečně výřečný v případě, že je nutné provést malé transformace dat (např. přepsání, což je ekvivalent operaci přiřazení), protože vyžaduje celou separátní aktivitu, která ve vizualizaci procesu zbytečně ruší, i když je její význam pro celkový proces margiální.

Dalším a větším problémem *XPM 1.0* je samotný *XDO*, který působí nekonzistentně a stojí za zvážení jeho výměna např. za štítek pro aktivity. Na takový štítek by bylo možné např. dvakrát poklepat, což by zpřístupnilo editaci *CSDDM 1.0* datového pohledu na aktivitu.

CSDDM 1.0 taktéž vizuálně postrádá jednoduchost v případě vlastností, které se měnit nebudou (např. oprávnění). Nahrazení stávajících zkratek za jednoduché a jasné symboly by významným způsobem přispělo k pochopení. Pro výběr či tvorbu vhodných symbolů by však bylo zapotřebí důkladné rešerše stávajících procesních notací, aby poté nedocházelo k záměnám.

XPM 1.0 je nadále zamýšlené jako producent vstupních specifikací pro projekt XPMA [2], který si klade za cíl z komplexních kombinovaných modelů vytvořit plně funkční, parametrizovaný systém pro různé platformy. XPMA by se poté měl zařadit mezi nejpokročilejší MDE nástroje současnosti a měl by mj. umožňovat i formální verifikaci a simulaci vytvořených modelů.

7.1 Zhodnocení výsledků

Výsledky této práce poukazují na nedostatečně holistický přístup k modelování komplexních systémů. Stávající dostupné datové modely systémů jsou zcela nedostatečně vybavené vzhledem ke dnešním požadavkům na distribuované nakládání s daty, kešování, paralelní přístupy atd.

Stávající procesní modely systémů jsou již dostatečně vyspělé na modelování reálných komplexních systémů a dokonce se podařilo oprostit se od objektově-centrického pojetí (které převážně stavělo na modelování

systémů po vzoru nepříliš vydařeného ekosystému Java), avšak stále nejsou moderní procesní modely (např. BPMN 2.0) dostatečně formální, aby bylo možné vždy jasně rozhodnout jakým způsobem modelovat danou situaci.

Nově představený model XPM 1.0 využívající CSDDM 1.0 pro modelování dat vyhovuje do značné míry holistickému pojetí komplexních systémů. CSDDM 1.0 byl původně zamýšlen jako vylepšený ekvivalent dokumentově orientovaných modelů, avšak po rozboru možností stávajících modelů ve spojitosti se zkušenostmi autora této práce byl model vystavěn zcela nanovo a se všemi vlastnostmi, které jsou nutné pro návrh moderních komplexních systémů či systémů s vysokou spolehlivostí.

Tato práce posunula pojetí Service Science blíže k uživateli a ukázala jak důležité je nesetrvávat v separovaných doménách výzkumu, nýbrž tyto vhodně a obezřetně propojovat. Služby jsou již dnes poháněny procesními systémy, ale od nynějška bude navíc možné výslednou hodnotu i samotného servisního systému spoluvytvářet s jeho odběratelem.

Bibliografie

[1] SEAM: A State-Entity-Activity-Model for a Well-Defined Workflow Development Methodology. Akhilesh Bajaj and Sudha Ram. 2001.

[2] XPMA - eXtensive Process Modeling and Automation, dost. online: https://github.com/dumblob/XPMA , cit. 4. 5. 2016.

[3] bpmn.io - A BPMN 2.0 rendering toolkit and web modeler., dost. online: https://bpmn.io/ , cit. 4. 5. 2016.

[4] SSADM foundation. Business Systems Development with SSADM. The Stationery Office. 2000. p. v. ISBN 0-11-330870-1.

[5] Business Rules Applied. Von Halle, Barbara. Wiley. 2001. ISBN 0-471-41293-7.

[6] Designing Data-Intensive Web Applications. Stefano Ceri, Piero Fraternali, Aldo Bongio, Marco Brambilla, Sara Comai, Maristella Matera. Morgan Kaufmann, USA. 2002. ISBN 978-1-55860-843-6.

[7] YAWL: Yet Another Workflow Language. W.M.P. van der Aalst and A.H.M. ter Hofstede. QUT Technical Report, FIT-TR-2002-06, Queensland University of Technology, Brisbane. 2002.

[8] Structured Design: Fundamentals of a Discipline of Computer Program and System Design. Edward Yourdon and Larry L. Constantine. Prentice-Hall, 1979 (Facsimile edition 1986). ISBN 0-13-854471-9

[9] The Entity-Relationship Model - Toward a Unified View of Data. Peter Chen. ACM Transactions on Database Systems 1 (1): 9–36. March 1976. doi:10.1145/320434.320440.

[10] Unified Modeling Language Reference Manual, The (2nd Edition). James Rumbaugh, Ivar Jacobson, Grady Booch. Pearson Higher Education. 2004. ISBN:0321245628

[11] OpenCASE– A Tool for Ontology-Centred Conceptual Modelling. Robert Pergl, Jakub Tůma. Springer Berlin Heidelberg. 2012. doi:10.1007/978-3-642-31069-0_42.

[12] Model-Driven Engineering. Douglas C. Schmidt. IEEE Computer, Vol. 39, No. 2, pp. 41-47. February 2006.

[13] The Process: Business Process Modeling using BPMN. Grosskopf, Decker and Weske. Meghan Kiffer Press. Feb 28, 2009. ISBN 978-0-929652-26-9.

[14] UML in practice. Petre, Marian. 35th International Conference on Software Engineering 18–26 pp. 722–731. May 2013.

[15] Kommunikation mit Automaten (Ph. D. thesis). Petri, Carl A.

University of Bonn. 1962.
[16] WebML: overcoming UML for web applications. Giorgio Sironi. dost. online: https://dzone.com/articles/webml-overcoming-uml-web , cit. 8. 5. 2016
[17] Semistructured Data. Peter Buneman. 1997
[18] The Rust Programming Language. dost. online https://www.rust-lang.org/ , cit. 8. 5. 2016
[19] The Definitive Guide to MongoDB: The NoSQL Database for Cloud and Desktop Computing (1st ed.). Hawkins, Tim; Plugge, Eelco; Membrey, Peter. Apress, p. 350. September 26, 2010. ISBN 978-1-4302-3051-9
[20] Bonitasoft Community BPM. dost. online https://sourceforge.net/projects/bonita/ , cit. 10. 5. 2016
[21] BizAgi Process Modeler. dost. online http://www.brothersoft.com/bizagi-process-modeler-109949.html , cit. 12. 5. 2016
[22] jBPM Console NG. dost. online https://salaboy.com/2013/10/11/using-the-jbpm-console-ng-hr-example/ , cit. 14. 5. 2016
[23] BPMN Process Monitoring using IBM Business Monitor 7.5. dost. online http://jay4tech.blogspot.cz/2012/11/bpmn-process-monitoring-using-ibm.html , cit. 18. 5. 2016
[24] Metasonic Suite 5.0. dost. online https://www.metasonic.de/en/metasonic-suite/5.0 , cit. 17. 5. 2016
[25] Definice procesu. Jaroslav Ráček. PV165 Procesní řízení. 2015
[26] Process Charts. Frank Bunker Gilbreth, Lillian Moller Gilbreth. American Society of Mechanical Engineers. 1921.
[27] Entity Relationship Diagram. dost. online https://www.edrawsoft.com/entity-relationship-diagrams.php , cit. 18. 5. 2016
[28] Electron. dost. online http://electron.atom.io/ , cit. 19. 5. 2016
[29] Semantic Versioning 2.0.0. dost. online http://semver.org/ , cit. 18. 5. 2016
[30] Disk Paxos. Leslie Lamport, Eli Gafni. Distributed Computing 16, pp. 1-20. 2003.
[31] Linearizability: A Correctness Condition for Concurrent Objects. Herlihy, Maurice P.; Wing, Jeannette M. ACM Transactions on Programming Languages and Systems 12 (3): 463–492. 1990.
[32] Cassandra - A Decentralized Structured Storage System. Lakshman, Avinash; Malik, Prashant. cs.cornell.edu. 15. 5. 2009.
[33] RIAK PRODUCTS. dost. online http://basho.com/products/#riak .

Cit. 25. 5. 2016.
[34] Bigtable: A Distributed Storage System for Structured Data. Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber. In OSDI. 2006.
[35] Datomic - The fully transactional, cloud-ready, distributed database. dost. online http://www.datomic.com/ , cit. 18. 5. 2016
[36] Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica. NSDI 2012. April 2012.
[37] MapReduce: Simplified data processing on large clusters. DEAN, J.; GHEMAWAT, S. In Proc. of the 6th OSDI, pp. 137–150. December 2004.
[38] RDF 1.1 Primer. Guus Schreiber, Yves Raimond, Frank Manola, Eric Miller, Brian McBride. https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/ , cit 18. 5. 2016
[39] Graph database. https://en.wikipedia.org/wiki/Graph_database , cit. 18. 5. 2016
[40] Key-value database. https://en.wikipedia.org/wiki/Key-value_database , cit. 18. 5. 2016

Seznam tabulek

Seznam obrázků

Přílohy

- xpma-case.zip

  V archívu xpma-case.zip je adresář s aplikací XPM Modeler, která implementuje část *XPM 1.0*. Pro spuštění aplikace je nutné archív rozbalit, přepnout se do adresáře xpma-case/ a spustit

  npm install

  CHROME_BIN=chromium grunt auto-build

  V hlavním okně aplikace je na některých platformách (zejména pokud je počítač pomalejší) nutné kliknout (někdy i vícekrát) na „Window“ a poté zvolit položku „Reload“ pro zobrazení hlavního okna XPM Modeler.