Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Grafické mapování systému

(Metodická tvorba a správa procesů za pomoci grafické notace)

Diplomová práce

Autor: Bc. Jan Nisler
Studijní obor: N6209 Systémové inženýrství a informatika

Vedoucí práce: doc. Ing. Hana Tomášková, Ph.D.

Hradec Králové Duben 2017

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 20.4.2017 Jan Nisler

Poděkování:

Děkuji vedoucí diplomové práce doc. Ing. Haně Tomáškové, Ph.D. za metodické vedení práce. Taktéž chci poděkovat vedení firmy za poskytnutí příležitosti pro tvorbu této práce. Dále chci poděkovat vedení fakulty za možnost studovat na FIM, kde jsem své znalosti prohloubil a nalezl jsem ve svém oboru zalíbení. Na závěr také chci poděkovat svým spolužákům, přátelům a rodině za pevné nervy se mnou samým.

Anotace

Název: Grafické mapování systému

Na efektivnost práce je kladen čím dál větší důraz. Je povinností každé firmy, aby s daným trendem udržovaly krok a tím i nabízely zákazníkům vždy kvalitní služby. Kde ovšem můžeme zefektivnit svou práci, když neznáme vnitřní procesy firmy? Jsou procesy důležité? Odpověď se může nalézat v grafickém mapování. Jaká je vhodná syntaxe? Je těžké se jí naučit? Může to ovládnout každý? Jak s tím pracovat, jak postupovat? Existuje nějaký jednoduchý postup, jak dosáhnout zmodelování celého systému? Mohou na daném projektu pracovat více uživatelů naráz? Tyto a další otázky se nacházejí v této práci o grafickém mapování systému.

Klíčová slova v této práci:

Procesní mapování, skladové hospodářství, více uživatelský přístup, subverison, informační systém, historie BPMN, význam BPMN, elementy BPMN, Business proces, sub-proces, UML Use case, scénář, Enterprise Architect

Annotation

Title: Graphic mapping of the system

On the effectiveness of the work is put more and more emphasis. It is the duty of each company to a given trend, keep pace and thereby offer customers with quality services. However, where we can improve our work, when we don't know internal processes of the company? Processes are important? The answer may be found in the graphical mapping. What is the appropriate syntax? Is it hard to learn it? Can everyone master it? How to work with it, how to proceed? Is there any simple way to achieve a comprehensive model of the system? Can work on the project several users at once? These and other questions are found in the work of the graphical mapping system.

Keywords in this work:

Process mapping, warehouse management, multi user access, subverison, information system, history BPMN, BPMN Meaning, elements of BPMN business process, sub-process, UML Use Case Scenario, Enterprise Architect

Obsah

Seznam obrázků

1 Úvod

Grafické mapování systému je v současné době jeden z nejdůležitějších faktorů, které by měly dnešní firmy řešit. Celkový pohled na systém, který nám poskytují grafické nástroje a jazyky, nám dovoluje daleko komplexnější pojetí systému a jeho okolí. V dřívějších dobách se každý vývojář musel zaobírat systémem pouze s využitím svých poznámek a vlastní představivosti. Vývoj analýzy systému tedy musel vyústit právě do sofistikovanějšího prostředí. Pouhý text už nestačí a pro efektivnější správu systému je zastaralým způsobem. Grafická notace nám už nabízí propracovanější přístup a dovoluje nám různé pohledy na jeden systém. Takový postup přerodu mezi textovou a grafickou formou vývoje systému je ovšem časově i cenově velmi náročný. Zapojení objektového modelování do praxe tak znamená velký a náročný krok vpřed.

Mnoho lidí však grafické mapování nevítá jako pomoc, ale jako přítěž. Tato přítěž dle jejich názoru spočívá v tom, že by se měli učit něčemu novému, přestože jejich starý přístup je dobrý a normálně funguje. Současně chápou jako zbytečnou zátěž nové postupy, které se musí naučit. To vše jsou však jen neopodstatněné výmluvy. Grafické mapování není nic neznámého a těžkého. Nemění naše dosavadní postupy při zpracování požadavků nebo procesního toku v systému a při implementaci do příslušného místa. Jediné co se mění, je forma zápisu. Z textové formy se pouze přechází do grafické. Aby se firmy dobře vypořádaly s takovými myšlenkami, je nutné dané postupy zavést v přísném režimu. Jedním z možných přístupů k zapojení grafického mapování do praxe je ten, že současné textové popisy systému se vymodelují do grafické podoby.

V případě firmy, která se zabývá **ERP**, je takový přístup ztížen. Celý systém se rozpadá do několika oblastí, které by měly samostatně fungovat, a tak je jejich mapování mírně ztíženo, protože nemodelujeme de facto jeden systém, ale hned několik. A právě v takové firmě zabývající se **ERP** byl tento postup aplikován. Na začátek celého přerodu z textové formy do grafické se použila oblast Skladového hospodářství. Tato oblast byla použita cíleně, jelikož je napojena na všechny ostatní oblasti ve větší míře, a proto je nejtěžší pro správné vymodelování. Právě tento projekt pro vymodelování Skladového hospodářství bude aplikován jako

šablona pro tvorbu ostatních oblastí a také pro namodelování celého systému podle jedné předlohy. Pro modelování byl v této práci využit **Case** nástroj **Enterprise Architect** a dvě syntaxe. Pro procesní mapování se použilo **BPMN**[1] a pro další mapování syntaxe **UML**[2]. Vybraná oblast, case nástroj a zvolené modely syntaxí budou popsány níže.

[1] BPMN neboli Business Process modeling and Notation, v dalších podkapitolách bude tato oblast vysvětlena
[2] UML neboli Unified Modeling Language, v dalších podkapitolách bude tato oblast vysvětlena

2 Cíl práce

Práce má několik cílů. Prvním a nejdůležitějším je zapojení grafického mapování do praktického života celého vývoje systému. Druhým cílem je ucelení poznatků o grafické syntaxi **BPMN** v jejich praktickém zapojení. Třetím cílem je vytvoření základní metodiky, podle které si může každý čtenář této práce vytvořit pohled na systém a tak si ucelit i poznatky o systému.

2.1 Teoretická část práce

Na začátku kapitoly jsou popsány funkční a procesní přístupy, které jsou jádrem firemní strukturu. V této části práce je také popsána vybraná oblast **ERP** systému, která bude vymodelována do grafické podoby. Oblast Skladového hospodářství je zpracována podrobněji tak, aby poskytovala ucelenou informaci o dané problematice. Obsahuje souhrn poznatků o všech zákoutích skladového hospodářství, které se mají vymodelovat. V druhé části je popsán význam podnikových procesů v grafickém prostředí a důvody nutnosti jejich mapování. V třetí části jsou popsány syntaxe, které jsou použity pro vymodelování. Jako poslední je popsán nástroj, ve kterém se tyto modely budou vytvářet, a jeho základní funkce.

2.2 Praktická část

Na začátku této kapitoly se představí samotné modelování, jako nástroj pro budoucí vývoj daného systému. Poté se popíše prostředí a princip modelování v nástroji **Enterprise Architect**. Jeho nastavení pro lepší ovládání a víceuživatelský přístup. Právě víceuživatelský přístup je potřeba při zpracování velkého projektu, na kterém se podílí mnoho vývojových pracovníků. Proto je tento přístup důležitý a bude zde popsán. V rámci toho je nutné vědět, jaký nástroj použít proti kolizím, a jaký přístup zvolit, abychom zamezili nechtěným vstupům do projektu. Další částí je popisem modelování procesní mapy. Procesní mapa bude vytvořena na příkladu jedné varianty od každého procesu v oblasti Skladového hospodářství. Z těchto procesních map bude vytvořen pohled na use case. Zde se podíváme, v jakém procesu, se jaký **use case** používá a k čemu byl vytvořen. Na závěr je uveden další možný diagram, který ukazuje systém z jiného

úhlu. Z vytvořených modelů se na závěr provede generování dokumentace v rámci naší šablony.

2.3 Závěr

V závěru práce budou výsledky shrnuty a vyhodnoceny. Podle přidané hodnoty bude navrženo kompletní přepracování celého systému do grafické podoby.

V příloze je uložena část dokumentace, která je vygenerována z case nástroje **Enterprise Architect**.

3 Teoretická část

3.1 Procesní přístup a Workflow

Podle autora Cardy[1] bylo dříve zvykem ve vedení firem používat funkční řízení. Tedy byla aplikována dělba práce, při které jsou procesy rozloženy na jednoduché činnosti, které jsou prováděny specializovanými lidmi. Tento přístup přivedl do světa hromadnou výrobu a specializaci. Kvůli změně charakteristiky trhu ovšem tento přístup není udržitelný. Do výroby už ve velké míře zasahují zákazníci a podle jejich potřeb se samotná výroba upravuje. Nastala tedy doba vlády zákazníka. Podle jeho potřeb je určován další průběh výroby produktů.

Taktéž podle Cardy[1] znamenal tedy funkční přístup řízení na základně hierarchicky organizované struktury. Jednotlivé úseky byly za svou práci zodpovědné a jejich efektivita se prokazovala ekonomickými výsledky za dané období. Kvůli tomuto rozdělení musely být vytvořeny kontrolní body, kdy se jednotlivá oddělení koordinovala a společně mohla určovat další průběh. V případě nefunkčního oddělení dochází k jeho reorganizaci tak, aby znovu plnilo svůj účel.

Zato procesní řízení je určeno, jak už název napovídá, procesním přístupem. Ve firmě jsou stanoveny hlavní **Business procesy**, které firmě přinášejí primární zisk. Na tyto hlavní procesy jsou napojeny procesy vedlejší, které napomáhají při tvorbě a dokončení hlavních procesů. Při formování základních procesů se vyskytují procesy, které nemají žádné opodstatnění, a proto je nutné takové procesy eliminovat.

Podnik může přejít na procesní styl řízení právě tehdy, když bude znát své primární procesy. Tyto závěry vycházejí od autora Fišera [2], když eliminuje všechny nepodstatné procesy a zaměří se na nedostatky svých hlavních procesů tak, aby mohl podnik svůj **Business** zlepšit. Ovšem takový stav ještě zdaleka nestačí. Pro hlubší pochopení svých procesů se firmy musí stát odborníky v odvětví, ve kterém pracují. Když znají dokonale prostředí a dané odvětví, mohou vzniknout procesy více flexibilní v přístupu k jejich zákazníkům. Vezměme si právě skladové hospodářství, které bude popsáno dále. Pro firmy zabývající se právě touto oblastí je nutností, aby znaly všechny možné kombinace ve skladovém hospodářství. Jelikož má každý zákazník nastavené jiné procesy ve skladování, je

nutností, aby firmy dokázaly obsáhnout veškeré možné kombinace. Tyto a další názory jsou i v článku tvorby autora této práce [12].

3.2 Skladové hospodářství

Podle autora Emmetta [3] hraje skladové hospodářství velkou roli ve výrobních firmách. Všechny oblasti firmy, od výroby po prodej, jsou závislé na skladovém hospodářství. Je to ze zjevného důvodu. Veškeré materiály, polotovary, výrobky a zboží (dále jen produkty) přechází právě přes skladové prostory. Skladové hospodářství, jako část podnikového logistického systému, se zaměřuje na tok materiálu. Přijímá produkty a ukládá je do skladu na specifické místo, odkud je následně vyskladní za účelem prodeje zákazníkům. Veškeré produkty, které přijdou a odejdou, musí být zaevidovány, aby bylo jasně definováno, kolik produktů projde přes sklady.

V dnešní uspěchané době je čas jedním z nejvýznamnějších kritérií z pohledu spotřebitele. Dodavatel, schopný dodat kvalitní výrobky v nejkratším čase, mnohdy vítězí nad konkurencí. Jak napovídá autor Nisler [12]. Aby výrobní firmy vytvořily podmínky pro dodávku svých výrobků zákazníkům v nejkratší možné lhůtě, musí si ze všeho nejdříve vyřešit právě skladové hospodářství. Ve skladech se zpracovává objednávka podle zákazníkových představ a odesílá prostřednictví některého druhu dopravy. Tento proces tedy zahrnuje většinu času od příjmu objednávky po její dodání. Ovšem právě skladové hospodářství je jedním z nejvíce měněných procesů. Každá firma si své skladové hospodářství vytváří sama a upravuje si i procesy v něm. Obecné vzory sice mohou pomoci, ale ve většině případů jsou upravovány dle konkrétních potřeb firmy. Jaké jsou obecné procesy a co to vlastně znamená skladové hospodářství, je rozebráno níže. Z výše uvedeného vyplývá, že **ERP** systémy musí být velmi flexibilní, aby mohly být pro každého zákazníka upraveny dle jeho specifických potřeb. Tyto závěry pochází od autorů Bigoš [5] a Jurové [6].

Ve skladovém hospodářství jsou tyto nejdůležitější rozhodovací akce:

- Řízení skladu
- Technologické prostředky skladu

- Rozsah a centralizace skladu
- Typ skladování
- Umístění skladu
- Hodnota zásob ve skladu

3.2.1 Funkce skladování

Je obtížné definovat jednotnou funkci skladového hospodářství v podniku, jelikož ve většině knih jsou funkce popsány pokaždé jinak. Po přečtení definic, které jsou obtížnější pro správné pochopení a převzetí z vlastní zkušenosti v oblasti skladu, byly vybrány, jako základní funkce podle autorů Sixta a Mačát [4]. Zde byly definovány 3 základní funkce skladování. V každé této funkci jsou procesy, které jsou ve skladovém hospodářství důležité. Všechny jsou v této práci vymodelovány a popsány tak, aby byly připraveny jako obecný pohled nebo základní **Warehouse Pattern** [3]na procesy v této oblasti.

Přesun produktů:

- Příjem zboží – v tomto procesu jsou definovány další dílčí sub-procesy, jako vyložení, vybalení, vstupní kontrola, kontrola dokumentace a aktualizace záznamů
- Uskladnění zboží – už v názvu jsou tyto procesy pro zaskladnění zásob na sklad, přesun na sklad a mezi sklady
- Kompletace zboží podle objednávky – přebalení objednávky podle požadavků zákazníka při vytvoření objednávky
- Překládka zboží – procesy určené pro přesun výrobků z místa příjmu do místa expedice
- Expedice zboží – zabalení a přesun objednávek pro dopravu. Kontrola vyskladněného zboží podle dodejky a aktualizace skladových zásob.

[3] Warehouse pattern – Standardy pro procesy ve Skladovém hospodářství

Uskladnění produktu:

- Přechodné uskladnění- uskladnění produktů pro doplňování základního stavu zásob
- Časově omezené uskladnění – zásoby spíše pro sezónní poptávku, kolísavou poptávku, úprava výrobků, spekulativní nákupy a pro zvláštní podmínky obchodu.

Přenos informací

- Tento proces upravuje hodnotu zásob, v přesunu, umístění zásob, vstupní a výstupní dodávky, zákazníků, personálu, a využití skladových prostor.

3.2.2 Význam skladování

Skladování zabezpečuje zaskladnění produktů, přesun produktu a vyskladnění produktu. Veškeré produkty uskladněné na skladu jsou zásoba v průběhu celého logistického procesu. Existuje mnoho druhů zásob. Příkladem je zásoba i na obaly, přepravní jednotky a jiné. V podstatě existují dva základní typy zásob, které je nutné evidovat na skladě. Těmto závěrům dochází mnoho autorů taktéž Mačát [4] a Emmett [3].

- Materiál, součástky a díly
- Hotové výrobky

Proč ale podniky udržují zásobu ve skladu?

- Snaha o dosažení úspor nákladů na přepravu
- Snaha o dosažení úspor ve výrobě
- Využití množstevních slev
- Udržení dodavatelského zdroje
- Podpora podnikové strategie v oblasti zákaznického servisu
- Reakce na měnící se podmínky na trhu
- Překlenutí časových a prostorových rozdílů, které existují mezi výrobcem a spotřebitelem

- Dosažení nejmenších celkových nákladů logistiky při současném udržení požadované úrovně zákaznického servisu
- Podpora programů **JIT**[4] u dodavatelů nebo zákazníků
- Snaha poskytnout zákazníkům komplexní sortiment produktů, nejen jednotlivé výrobky
- Dočasné uskladnění materiálů, které mají být zlikvidovány nebo recyklovány

3.2.3 Vliv skladování

Jak už bylo zmíněno výše, skladování se prolíná do většiny oblastí výrobních podniků. Má na ostatní oblasti vliv zejména zásobou a svými procesy. Jedním z nejdůležitějších propojení je propojení s výrobou. Podle autora Mačáta [4] se jednotlivé vlivy dělí na následující vztahy.

Vztah mezi skladováním a výrobou

Při minimalizaci objemu zásob podniky vyrábějí podle poptávky po daném produktu. Ovšem při změně produktu musí dojít ke změně výrobních linek, a tak vzniká prodleva, kdy z výrobních linek nesjíždí daný výrobek. Tím dochází ke ztrátám v podobě nevyužitých příležitostí k prodeji. Cena produktu je pak vyšší z důvodu neustálých změn výrobních linek v závislosti na poptávce. Na druhou stranu tento způsob výroby nemá skoro žádné náklady na skladování a jejich stav zásob zůstává minimální. Pokud se však při každé změně výrobních linek vyrábí více produktu, než je potřeba, vzniká tím zásoba a tím se snižuje pravděpodobnost promarněných šancí na prodej. Vytvořená zásoba může tyto potencionální poptávky v době přenastaven výrobních linek pokrýt, a tím místo ztráty přichází zisk. Tato varianta však s sebou nese větší náklady na skladování a nutnost dvojí manipulace s produkty. Každá varianta má tedy své pro a proti, a se skladováním se pracuje v menším či větším rozsahu. Každý podnik si musí spočítat, jaká varianta se mu vyplatí a přinese větší zisky. V množstevní výrobě je také ta výhoda, že podnik může zavést množstevní slevu pro své věrné zákazníky a při nižších nákladech na výrobu může mít větší zisky. Taktéž při dopravě zákazníkovi, pokud

[4] Jit – Just in time metoda, bude vysvětlena níže

ji výrobní podnik platí, může dosáhnout snížení nákladů díky přepravenému množství na jednu dopravu. Vše je nutné propočítat a vyhodnotit podle časového intervalu, za který může podnik dopravit svým zákazníkům produkt.

Vztah mezi skladováním a přepravou

Úspory nákladů na přepravu se mohou snížit při vstupních a výstupních procesech a také při vnitropodnikových přesunech produktů. Tyto procesy se musí co nejefektivněji navrhnout, aby manipulace s produkty nebyla nikterak složitá a co nejméně nutná. Při zvětšeném přesunu produktu mezi sklady nebo při převozu na jiné umístění dochází k opotřebení obalu produktu a tím může dojít i k nechtěnému znehodnocení materiálu. Zkušenosti potvrzují, že při přemrštěném přemisťování produktu se zvyšuje pravděpodobnost vzniku chyb i pravděpodobnost zničení produktu v důsledku nepatřičného zacházení.

Vztah skladování a zákaznického servisu

Pro efektivnější plnění zákaznických potřeb je dobré, aby podnik vybudoval řadu lokálních skladů, díky kterým lze minimalizovat náklady na přepravu výrobku zákazníkům. Při přijetí objednávky může podnik rychleji zareagovat na zákazníkovi potřeby a vyexpedovat dříve jeho objednávku. Tak může snížit náklady na přepravu zákazníkům a rychleji reagovat na jejich potřeby.

Vztah skladování a logistiky

Skladování není vždy využíváno pro minimalizaci nákladů na skladování a zásobování zákazníků. Neexistuje obecný postup, kterým se podnik má řídit, aby si nastavil své skladovací procesy. Jelikož každý podnik je víceméně vybudován podle individuálních pravidel, není možné najít dva identické podniky, mezi kterými by neexistoval vzájemný vztah. Každý podnik se proto musí řídit několika faktory, které ovlivňují strategii skladování. Tyto faktory jsou podle Mačát [4], ale také podle autora Jůrová[6] tyto:

- **Odvětví –** každé odvětví má na skladování jiné nároky, například potravinářský průmysl musí dbát na datum spotřeby potravin a expedovat je

jinak, než například v hutním průmyslu. Dalším problémem je, že každé odvětví musí splňovat jiné skladové podmínky. Jak bylo zmíněno, potravinářský průmysl musí mít sklady uzpůsobené pro uchování potravin, aby jejich kvalita byla zajištěna.

- **Podniková strategie –** jak už bylo zmíněno, každý podnik má svou vlastní strategii, kterou plní, právě proto musí být skladování uzpůsobeno přesně podle strategie celého podniku.
- **Charakter výrobku –** Tento faktor je velmi důležitý. Je to z hlediska rozměrů daného výrobku, výrobní řady, šarže, možnosti substituce a míry zastarávání. Je to z pochopitelných důvodů, jelikož čím větší máme výrobky, tím jsou vyšší nároky na přípravu skladových prostor, kam mohou být výrobky uskladněny. Z druhé strany pokud chceme hledět na výrobní řady, musíme mít stejné produkty oddělené podle této řady, jelikož když bude jedna řada špatná, víme, z jaké řady pochází a tak snadněji můžeme opravit zbývající produkty z téže výrobní řady.
- **Konkurence –** Tento faktor je také velmi důležitý. Konkurence může být veliká, a abychom si udrželi své postavení na trhu, musíme být konkurenceschopní. Například čas hraje při udržení zákazníků velikou roli. Pokud můžeme uspokojit zákazníkovi potřeby v relativně krátkém čase, je pravděpodobné, že si zákazník objedná další produkty právě u nás a nepřejde ke konkurenci.
- **Sezónní poptávky –** Pokud víme, který produkt se více prodává v jakém období, můžeme tomu uzpůsobit naši výrobu i styl uskladnění. Produkty můžeme následně uskladňovat podle míry jejich prodeje.
- **Ekonomické podmínky –** Tyto podmínky jsou určeny situací na trhu, jestli je po daném produktu poptávka, jak nám změna poptávky mění cenu daného produktu.
- **Dostupnost kapitálu -** Tento faktor je podnikový. Pokud je na tom podnik ekonomicky dobře, může vyrábět nad rámec poptávky a tím si vytvářet zásobu produktu. Pokud ovšem podnik přežívá a má menší zisk, nemůže vytvářet větší zásobu.

- **Použití přístupů JIT** (**Just in Time**) **–** Tyto procesy nemusejí být pouze **JIT**, ale také jiné logistické technologie.
- **Použití výrobního procesu -** Tento faktor může dopomoci k rozhodování podle typu výroby, které v podniku je. Pokud se vyrábí přesně pro potřeby zákazníků, musí docházet k neustálému přenastavení strojů na jiný produkt, který je právě více žádán.

3.2.4 Moderní přístupy v řízení zásob

Abychom měli jasnou představu, jak skladování působí v podnicích, musíme taktéž vědět, jak jsou tyto procesy provázány v běžném chodu podniku. Tyto přístupy řeší, jak a kdy se se zásobami manipuluje, jaké operace probíhají a jaká vybavenost skladu je zde zastoupena. Jsou to metody pro optimální fungování skladování podle autorů Emmet[3] a Mačát[4].

Kanban

Bezzásobová technologie, která byla vyvinuta japonskou společností Toyota Motors a rychle se rozšířila do výrobních podniků celého světa. Někdy je známa jako **Toyota Production Systems** (**TPS**). Tato metoda se nejlépe používá ve velkosériové výrobě s neustálým prodejem, kde je jednosměrný tok materiálu, výrobní operace lze snadno sladit a nedochází k velkým změnám požadavků na finální produkt. Je to právě v takové výrobě, kde jsou produkty podniku jednolité, bez zasahování zákaznických potřeb. Kanban vychází z následujících principů:

- Fungují zde samo řídící regulační okruhy, které tvoří dvojice článků vzájemně propojené na základě „**Pull principu**“
- Objednacím množstvím je zde obsah jednoho přepravního prostředku nebo násobků, plně naplněných vždy konstantním množstvím
- Dodavatel ručí vždy za kvalitu dodávaného množství a objednatel ručí za přijetí objednaného produktu.
- Kapacity dodavatele a odběratele jsou v rovnováze
- Spotřeba materiálu je bez výkyvů stálá
- Nevytváří se žádná zásoba u obou stran dodání

Just in Time (JIT)

Tato metoda je nejznámější logistickou technologií. Vznikla počátkem 80. let v Japonsku a USA. Jde o způsob uspokojování poptávky po určitém materiálu ve výrobě nebo hotovém výrobku v distribučním řetězci v přesně dohodnutých a dodržovaných termínech, dle potřeb odebírajících článků. Jednoduše můžeme říct, že metoda **JIT** je rozšířením metody **Kanban**, protože propojuje nákup, výrobu a logistiku. Metoda **JIT** se zaměřuje na problémové úseky výroby produktu, možnosti, kde daný proces zlepšit, vymazat nepotřebné činnosti a celý proces zefektivnit. Metoda nám tedy říká, jak vylepšit procesy, aby se zamezilo ztrátám. Je ovšem náročná pro implementaci do systému, jelikož zahrnuje všechny články od dodavatele přes distributora až k odběrateli. Všechny tyto články musí být ve vzájemné shodě a spolupráci. Právě kvůli tomuto obtížnému řízení je dobré zvážit, zdali je tato metoda vhodná pro využití v podnicích či nikoliv.

Při uplatňování této metody dochází k růstu nákladů na přepravu se snižováním přepraveného množství zboží při jedné dodávce a se zvyšováním celkové rychlosti přepravy. Dále bude docházet k poklesu nákladů na skladování v závislosti na snižování přepraveného množství zboží při jedné dodávce a k vázanosti kapitálu v závislosti na růstu rychlosti přepravy.

Při zapojení metody **JIT** do systému je odběratel uvažován jako dominantní článek a dodavatel se musí přizpůsobit požadavkům svých zákazníků. Dodavatel tak garantuje požadovanou kvalitu dodávky a poskytuje informace potřebné pro plánování a operativní řízení. Dále musí být přeprava svěřena kvalitnímu dopravci.

Hub and Spoke

Tato metoda předpokládá slučování menších zásilek do větších celků, které jsou po přepravě do místa určení opět rozděleny do menších celků. Celá metoda tedy tkví v tom, že objednávky se sdruží do většího celku a tento celek se pomocí železniční, vodní, kamionové či letecké dopravy přepraví na místo určení. Zde se poté rozčlení zpátky na menší objednávky, které už lze rozvézt zákazníkům pomocí menších dopravních prostředků. Tímto se snižují náklady na přepravu. Sdružovat se objednávky mohou do některého typu přepravní jednotky s větší kapacitou, například kontejnery.

Tato technologie si v porovnání s **JIT** dokáže poradit i s menšími objednávkami, které mohou být častější. To vše ekologičtějším a levnějším způsobem. Při dobře zvládnuté organizaci se daří při krátkodobém skladování části produktu v logistických centrech zásobovat odběratele pravidelně malými dodávkami obdobně jako u metody **JIT**. Výhodou této metody jsou nižší náklady na dopravu, odlehčení dopravních komunikací a ekologická šetrnost. Na druhou stranu je tato investice nákladnější a je víceméně použitelná na delší dopravní vzdálenost.

3.2.5 Sklad a vybavení

Sklad je fyzické místo, kde se provádí skladové činnosti. Tedy je to budova, místnost, umístění, kde se provádí přejímka produktu, uskladnění, vyskladnění, expedice produktu a přidružené manipulační operace. Rozvržení celého skladu je jedním z nejdůležitějších kritérií, které je při skladování nutné definovat. Základní charakteristika skladu je tato. Čtyři stěny, nejčastěji z betonu, dřeva či slitiny kovu, izolované od elektrického vedení, které je součástí celého skladu. Do tohoto uspořádání patří i vybavení celého skladu:

Vybavení skladu

Ve skladovém zařízení existuje několik typů vybavení. Jedním z nich je vybavení manipulační. Toto vybavení slouží k přesunu produktu z jednoho místa na druhé. Dalším vybavením je vybavení pro uskladňování. Tímto vybavením jsou myšleny regály či police. Tato vybavení musí být kompatibilní, jelikož nelze kombinovat libovolné typy vybavení mezi sebou.

Aktivní prvky

Aktivní prvky jsou určeny pro netechnologické operace s pasivními prvky. Aktivní prvky tedy převádějí jednotlivé produkty z jednoho místa na jiné místo určení, ale slouží taktéž pro operace balení, tvorby a rozebíraní manipulačních a přepravních jednotek, nakládky přepravy, překládky, vykládky, uskladňování, kompletace, kontroly, sledování či identifikace, ale i sběr informací, přenos informací a uchování informací. Tyto prvky jsou velmi členěné, a proto jsou zde pouze vypsány. Aktivní prvky se dělí na manipulační prostředky a zařízení a

dopravní prostředky. Tyto aspekty jsou dále rozebrány i u autorů Emmet [3] a Mačát [4].

Pasivní prvky

Pasivními prvky je myšlen materiál, přepravní prostředky, obaly, odpad a informace. Jsou to prvky, se kterými lze manipulovat, přepravovat nebo skladovatelné kusy, jednotky nebo zásilky. Jsou to materiály, tedy prostředky, které jsou určené pro produkci, výrobu a služby. Další jsou manipulační prostředky. To znamená přepravky různých velikostí určené pro přesun produktu, pro jejich zabezpečení proti ztrátě kvality. Ostatní jsou například palety, roltejnery[5], přepravníky, kontejnery, a výměnné nástavby. Dalšími pasivními prvky jsou obaly. Tyto prvky jsou určeny pro tři základní funkce. Pro přepravu, pro ochranu a pro informaci, proto, aby byl produkt dobře chráněn při manipulaci. Taktéž obaly mohou být nositelé informací, jelikož v nich může být uskladněno veliké množství produktů menších rozměrů. Tyto informace mohu být na etiketě umístěny ve formě kódu, čárového či QR.

3.2.6 Velikost a počet skladů

Pro objasnění, proč mít skladové hospodářství nutně v podnicích, je neméně důležitá otázka, jak velký sklad a počet skladů má mít každý z podniků. Právě vedení podniku musí řešit tyto dvě otázky. Na každou otázku si však musí odpovědět zvlášť, jelikož tyto dva faktory jsou mezi sebou obvykle v nepřímé úměře. Částečnou odpověď nám mohou dát rovněž autoři Bigoš [5] a Jurová [6]

Velikost skladu

Jak má být sklad velký, určuje několik faktorů, jako je například druh zásob, velikost zásob a jiné. Pro měření velikosti skladu je směrodatná velikost skladové plochy nebo objem skladového prostoru. Většina skladů používá při inzerci a propagaci svých zařízení stále ještě informace udávající skladovou plochu – v m^2.

Jelikož tento údaj nebere v potaz vertikální uskladnění, používá se v dnešní době měření v m^3. Toto trojrozměrné měření skladu nám udává většinu

[5] Roltejner – druh přepravního prostředku ve skladech, založeno na principu čtyřkolového podvozku

skladového prostoru, který má podnik k dispozici. Při úvahách o velikosti skladu ale hrají roli ještě další faktory:

- Úroveň zákaznického servisu
- Velikost trhu
- Počet skladových produktů
- Velikost skladových produktů
- Používání systému manipulace s materiálem
- Typ používaného skladu
- Pohyb zboží ve skladu
- Celková doba výroby produktu
- Velikost kancelářských prostor v rámci skladu

Tyto faktory jsou popsány i v podobných významech výše, kdy se řešil vztah skladování a logistiky. Stručné shrnutí těchto faktorů je možné podle autora Mačát [4] toto:

Úroveň zákaznického servisu závisí na potřebách samotného zákazníka. Pokud potřebuje vyšší množství produktu, musí se vyřešit i množství uskladňovacích prostor. Když se navýší velikost trhu a zvýší se i počet zákazníků podniku, musí být podnik schopen na tento stav reagovat a mít skladovací prostory právě pro tento stav. Na velikosti skladu také závisí i počet produktů, které daný podnik vyhotovuje. Pokud je podnik s větším množství druhů produktu, je logické, že jeho sklady musí být větší, aby byly všechny produkty řádně uskladněny. Pochopitelně záleží i na velikosti vyráběných produktů. Při výrobě větších kusů musí být uzpůsobeny i skladové prostory pro takové produkty a manipulaci s nimi. V malém skladu je větší pravděpodobnost, že při manipulaci s většími produkty dojde k jejich poškození, či poškození samotného skladu. Dalším faktorem, který bude řešen níže, je používaný systém manipulace s produkty. Navíc s typem používaného skladu, který bude taktéž popsán níže, jsou tyto faktory velmi důležité. Pro určení velikosti skladu je také nutné zodpovědět otázky pohybu produktu po skladu. Pokud se produkt pohybuje v přemrštěné míře po skladových plochách, musí k tomu být uzpůsobeny i skladové manipulační prostory. Navíc se

nesmí počítat pouze s užitkovou skladovou plochou a uskladňovacími prostory. Ve skladových částech musí být i kancelářské prostory, buď přímo v něm či blízko k nim přidružené.

Počet skladů

Při otázce kolik má mít podnik skladů, záleží na 4 základních faktorech. Náklady související se ztrátou prodejní příležitosti, náklady na zásoby, náklady na skladování a náklady na přepravu.

Náklady související se ztrátou prodejní příležitosti

Tyto náklady se liší od odvětví k odvětví. Když podnik ztratí prodejní příležitost, tratí na zisku a čas vynaložený na přípravu k prodeji je jen promarněným časem. S přibývajícím počtem skladů tyto náklady rostou a je nutné, aby si každý podnik mohl propočítat případná rizika, která mohou nastat podle autorů Bigoš[5] a Emmett[3].

Náklady na zásoby

Je logické, že při zvětšování počtu skladů se zvyšují i náklady na zásoby. Při lokálních skladech navíc udržuje podnik minimální hodnotu všech svých produktů, aby v případě potřeby mohly být expedovány v co nejkratším čase zákazníkům. Tento princip však staví podnik do situace, že na každém skladu musí mít podobný systém uskladnění, aby nedocházelo ke zmatkům a nejasnostem.

Náklady na skladování

V případě zvětšování počtu skladů se taktéž zvyšují náklady na skladování, jelikož musí být v každém skladu přepravní prostředky a personál. S tím souvisí náklady na servis strojů a mzdy pracovníků. Určitým způsobem to podniky mohou řešit najímáním veřejných skladů, které poskytují množstevní slevy na uskladnění produktu, ovšem za předpokladu, že má podnik pronajaté sklady od té samé firmy, která tyto veřejné sklady vlastní.

Náklady na přepravu

Při větším počtu skladů se zvyšuje množství nákladů vynaložených na vstupní a výstupní přepravu. Tento jev ovšem nastává až po situaci, kdy náklady na přepravu prvně klesají a při zvýšení množství skladů se náklady zvětšují. Každý podnik si proto musí jasně definovat nejoptimálnější řešení, aby neplatil více.

3.2.7 Funkce skladu

Sklady jako fyzické prostory pro manipulační procesy s produkty slouží pro sladění rozdílně dimenzovaných toků. Podle autora Mačáta[4].

Vyrovnávací funkce

V případě, že je materiálu nedostatek k výrobě, musí dojít ke zvýšení stavu zásob na výrobě, a tím i k optimálnímu stavu výroby.

Zabezpečovací funkce

Sklad musí efektivně dodávat produkty na výrobu, uskladňovat produkty a musí být připraven k okamžitému vyskladnění produktu z důvodu nečekané objednávky zákazníka. Míra flexibility je charakteristikou skladu.

Kompletační funkce

Na skladě se taktéž kompletují objednávky. Podle potřeb musí být objednávka řádně zabalena do předepsaného obalu a musí být umístěna na specifické přepravní jednotce.

Spekulační funkce

Vyplývá z očekávaných zvyšování cen na zásobovacích a odbytových trzích.

Zušlechťovací funkce

Zaměřena na jakostní změny uskladňovaného produktu. Každý produkt má své vlastnosti a přímo ve skladech musí být prostředí uzpůsobeno podle druhu produktu. Sklady pro potraviny musí vykazovat jiné vlastnosti než sklady pro hutní průmysl.

3.2.8 Druhy skladu

Podniky uskladňují produkty na předem nachystané sklady. Každý podnik však vytváří svou vlastní síť typů skladů, do kterých podle jejich specifik ukládá dané produkty. Sklady je možné dělit podle různých kritérií, zde jsou popsány 3 oblasti dělení. Tyto oblasti byly vybrány jako obecné rozdělení skladů podle procesů, které se v daném skladu provádí.

Podle vlastnictví

Jak již bylo zmíněno výše, podnik může mít svoje sklady či pronajaté sklady, tedy může je vlastnit či nikoliv. Toto rozhodnutí je založeno na principu Make or Buy.

- **Soukromé sklady**

Tento typ skladu je sklad ve vlastnictví samotného podniku a veškeré vnitřní procesy jsou plně v kompetenci daného podniku. Podnik tedy platí za technologickou vybavenost skladu, za manipulační jednotky, pracovníky, systém a uskladňovací jednotky. Soukromé sklady mají výhody v podobě nezávislosti na dalším subjektu, flexibilitě vlastního řízení a otevřené nákladovosti, tzn., že podnik vidí přesně, za jaké činnosti musí platit při správě skladu. Co se ale týká soukromých skladů, jsou levnější, nežli sklady veřejné. V publikacích se uvádí snížení nákladů v soukromém skladu oproti veřejnému okolo 18-20 %, za podmínky, že je sklad využíván minimálně na 80%. Další velkou výhodou soukromého skladu je marketing. Vlastní sklad v podniku působí na zákazníky dojmem, že firma je na tom ekonomicky velmi dobře, tudíž si může dovolit vlastní sklady a je schopna reagovat na změny. Na druhou stranu nejsou soukromé sklady schopny měnit svou kapacitu podle měnící se poptávky. Podnik tedy nemůže růst za své hranice a musí si smluvně zajistit další skladové prostory.

- **Veřejné sklady**

Podnik může využívat také veřejné sklady, kde je možné na určitý časový interval zaskladnit svoje produkty v kombinaci s nabízenými službami firmy, která dané sklady spravuje. Veřejné sklady mohou být pro smíšené zboží, chladírenské sklady, sklady pro zboží se speciálními požadavky, sklady pro zboží pod celní uzávěrou a sklady pro domácí potřeby a nábytek. Veřejné sklady poskytují podnikům finanční flexibilitu a benefity z úspor ze zvýšené výroby. Podniky tak mohou vyrábět více, než by mohly, kdyby využívaly pouze své sklady. V oblasti financování těchto skladů je otázkou, jaké tarify by bylo nutné platit. Ale podnik už nehradí mzdové náklady, nutné investice do nových technologií, daně za pozemek, budování nových skladů a samotnou výbavu skladu. Veřejné sklady může podnik využívat na kratší čas, či lze upravit dohodu dle měnících se podmínek na trhu. Další výhodou je rychlá odezva při rychlých transakcích pro odběratele či samotného podniku. Nevýhodou těchto skladů je komunikační problém. Každý podnik zabývající se sklady nemusí poskytovat všechny informace o skladových

procesech a samotný systém nemusí přizpůsobovat pro každého zákazníka. Podniky, které daný veřejný sklad využívají, musí s tímto omezením počítat.

- **Smluvní sklady**

Tyto sklady kombinují oba předchozí typy skladů. Dlouhodobé vztahy se sdíleným rizikem snižují náklady a přinášejí větší výhody v podobě využívání specializovaných znalostí, sdílení pracovních a manipulačních vybavení a informačních zdrojů.

Podle funkce

Sklady jsou podle tohoto kritéria rozděleny podle funkce, kterou plní.

- **Obchodní sklady.**

Sklady, ve kterých je větší množství dodavatelů i odběratelů Funkcí obchodních skladů je především změna sortimentu.

- **Zásobovací sklady**

Tyto sklady jsou převážně budovány pro dostatečnou zásobu výroby. Na tyto sklady jsou převážně dodávány produkty, které jsou určeny pro další zpracování ve výrobě.

- **Celní sklady**

Sklady jsou určené pro zaskladňování produktu například tabáku či alkoholického rázu. Dokud nejsou tyto produkty distribuovány na trh, má nad nimi stát výhradní kontrolu. V tomto okamžiku musí dodavatel zaplatit celní poplatek. Výhodou je fakt, že zaplatit clo může dodavatel až v okamžiku, kdy se daný produkt prodá.

- **Tranzitní sklady**

Tyto sklady jsou určeny pro manipulaci velkého množství produktu. Například přístavy či letiště. V těchto skladech dochází při příjmu k okamžitému

rozdělení produktů podle zákazníků, následně je zabezpečeno naložení na další dopravní prostředky a odeslání na místo určení.

- **Cross – docking**

Tento sklad převážně slouží jako distribuční směšovací centrum, kde dochází k okamžitému předávání produktu. Produkty jsou do tohoto skladu převáženy ve velkém množství. Okamžitě při přijetí dochází ke kastomizaci produktu do jednotlivých zásilek určitých zákazníků. Produkty v tomto skladu se nezdrží více jak 24 hodin a jsou bezprostředně po kastomizaci expedovány.

- **Konsignační sklady**

Konsignační sklady jsou v kompetenci dodavatele a odběratele. Tyto sklady vznikají na základě dohody mezi dodavatelem a odběratelem. Odběratel si zásoby produktů, které odebírá od daného dodavatele, nechává zaskladněné na skladu dodavatele a platí pouze za ty produkty, které si odebral. Většinou je dodavatel vytváří blíže k odběratelům, aby byly zákaznické potřeby uspokojeny v co nejkratším časovém intervalu.

Podle technologického vybavení skladu

Tento typ rozdělení skladu spíše rozděluje sklady podle vybavenosti skladu a zavedených procesů. Každý z níže uvedených typů řeší skladové procesy jinak než druhý. Tyto procesy a jejich varianty jsou rozepsány a zmodelovány v praktické části.

- **Plně automatizované sklady**

Tyto sklady jsou plně zautomatizovány, takže veškeré procesy, v nich prováděné, jsou prováděny jen automatizovaně. V tomto skladu je zastoupení lidského faktoru minimalizováno.

- **Automatizované sklady**

Část procesu pro manipulaci produktu je zajištěna automatizovaně, například ukládání manipulačních jednotek a jejich vyexpedování v rámci objednávky.

- **Mechanizované sklady**

Ve skladech jsou některé procesy zajišťovány pomocí mechanizačních zařízení. V procesech však nejsou mechanizační prostředky zastoupeny stoprocentně. Velký podíl na provádění procesů má i lidský faktor.

- **Vysoce mechanizované sklady**

Sklad založený na progresivních technologiích s prvky automatizace, přičemž ve všech procesech je taktéž zastoupen i lidský faktor. Tento typ skladů je nejrozšířenějším typem a je nejvíce používán v evropských zemích.

- **Ruční sklady**

Procesy ve skladech jsou realizovány pouze s využitím manuálního řízení člověka.

3.2.9 Trendy ve skladování

Jaké trendy ve skladování jsou, to nám napovídá autor Mačát[4]. S ohledem na zvyšování úrovně zákaznického servisu musí dodavatel klást větší důraz na samotné skladování. V rámci vyřizování objednávky je většina manipulačních procesů koncentrována právě ve skladovém hospodářství. Čas hraje v dodávkách čím dál větší roli. Kdo dokáže kvalitně a rychle uspokojit zákazníkovi potřeby, zpravidla získá zákazníky. Navíc, když vezmeme v potaz nynější charakter objednávek (menší ale častější), musí podnik počítat s větším tlakem právě na skladové procesy. Probíhají kvantitativní procesní toky, a pokud na to není podnik řádně připraven, může to vést ke zhroucení skladového systému.

Východiskem z této situace může být centralizování skladů, kdy dochází k seskupování podobného druhu sortimentu, což vede k větší rychlosti vychystání a

menším úsporám. Dalším řešením může být zvyšování technologické úrovně skladu. Dnešní trend je takový, že si zákazníci objednávají produkty přesně na míru jejich potřebám. Tak vznikají specifické produkty, které je nutné zaskladnit a následně po kompletaci i odeslat. V důsledku toho vzniká větší důraz na snižování zásob, jelikož specifika objednávky jsou dodavateli známá až ve chvíli objednávky. Tento postup se skvěle hodí na metodu **JIT**. Tato metoda klade velký důraz na včasné vychystání a sledování produktu od výroby podle šarže až po expedici k zákazníkovi. Tak může dodavatel sledovat celou zásilku od naložení na dopravní prostředek na skladě až po jeho dodání k zákazníkovi. Má tak celou trasu dobře zmapovanou a částečně tento faktor slouží i ke sledování celkových nákladů na přesun produktu k zákazníkovi.

Optimalizační přístup

Tento přístup patří mezi strategie skladování. Při malé zásobě nemusí mít podnik velký sklad, ale pouze menší sklad, kde jsou zaskladněny produkty. Pro manipulaci s produkty postačí několik málo manipulačních bodů. Tak dochází ke snižování nákladů a jejich přesnějšímu definování. Zde však vyvstávají otázky v pohledu na metodu ukládání produktu do skladů. Tyto aspekty jsou rozšířeny od autora Emmet [3]

- **Metoda pevného ukládání**

Ke každému produktu je přiřazeno určité místo, které je systémem zaevidováno pro daný produkt. Tato metoda dopomáhá k lepšímu náhledu do skaldových zásob a personál tak ví, kde je jaký produkt uskladněn. Nevýhodou je neefektivní využívání skladových kapacit. Tyto údaje dále rozebírá autor Jurová [6].

- **Metoda záměnného ukládání**

Každý produkt může být uložen do libovolného skladového umístění. Protože se zásoba produktu zpravidla doplňuje postupně pro maximální celkovou zásobu ve skladu, postačí menší kapacita než při pevném ukládání. Menší sklad

také snižuje délku pohybu mezi ukládacím místem a předávacím uzlem. Tato metoda však nezohledňuje frekvenci prodeje jednotlivých produktů, a tak může dojít k situaci, kdy je v popředí zaskladněn produkt, který je jen zřídka prodáván. Brání tak ukládání frekventovanějšího produktu a celková doba procesu ve skladu se může tímto prodloužit. Jako u přechozího je tento aspekt více rozebrán i u autora Emmett [3]

- **Metoda skladových zón**

Tato metoda slouží k tvorbě skladových zón. Do jednotlivých zón jsou ukládány produkty se stejnou či podobnou frekvencí prodeje. Do popředí skladu jsou ukládány produkty, které jsou více prodávány a v nejvzdálenější části skladu jsou produkty, které jsou jen zřídka prodávány. Ukládání do jednotlivých zón je řešeno pomocí záměnného ukládání. Takto se průměrná doba procesu snižuje, avšak celková kapacita skladu musí být větší nežli při metodě záměnného ukládání, protože se musí ukládání dimenzovat pro špičkovou zásobu produktu v každé ze zón ve skladu.

- **Metoda dynamické zóny**

Tato metoda spočívá v dynamické klasifikaci jednotlivých produktů a dynamickém rozvržení zón. Metoda byla vytvořena kvůli neustálým změnám ve strategii řízení zásob a strategii velikosti objednávky. Jednotlivé položky mohou najednou vyhotovovat klasifikačním kritériím jiných skladových zón, vznikají nové produkty a jiné jsou rušeny. Kvůli tomuto se příslušnost produktu k zónám periodicky přizpůsobuje aktuální situaci a rámcovým podmínkám.

- **Metoda připraveného vyskladňování**

Metoda řeší problémy předešlých metod. Zde se řeší co nejmenší časový úsek mezi příjmem příkazu k vychystání a časem odeslání. Proto jsou produkty přeskladňovány podle času jejich vyskladnění. Tak se zkracuje časový interval procesu, ale nároky na kvantitu procesu se zvyšují. Předpokládá se ovšem existence prostojových časů pro manipulační zařízení.

- **Metoda předvídajícího uskladnění**

Tato metoda řeší některé nedostatky předchozích metod. Metoda předpokládá zaskladňování produktu na volná místa v závislosti na době předpokládaného vyskladnění položky. Produkty, které budou dříve vyskladněny, jsou umisťovány na volná místa v popředí skladu. Vždy se řeší co nejideálnější místo pro uložení produktu. Cílem této metody je co nejkratší doba pro manipulaci produktu při vyskladnění a zároveň co nejmenší počet manipulačních procesů v rámci jednoho produktu,

3.2.10 Časté chyby při skladování

Všechny výše zmíněné aspekty si každý podnik musí uvědomit a na jejich základě si musí vybudovat svou infrastrukturu skladového hospodářství. Důležitou součástí skladového hospodářství je i řízení zásob. Jak velký je význam skladování pro každý podnik je zřejmé, stejně tak jako skutečnost, že na sestavení optimálních skladových principů by měl být kladen velký důraz. Při nesprávném použití jednoho či druhého nástroje může dojít k obrovským ztrátám v případě, že se ukáže, že daný princip nemá v daném podniku žádnou užitnou hodnotu. Dílčí chyby, kterých se může každý podnik dopustit. Tyto chyby jsou vzaty z praxe od autora Mačát [4] a Jurová[6]:

- Přebytečná a nadměrná manipulace s produkty
- Nízké či žádné efektivní využití skladových ploch a prostoru
- Nadměrné náklady na údržbu a výpadky kvůli zastaralým zařízením
- Neefektivní procesy pro skladové pohyby produktů
- Špatné využití výpočetních technologií při rutinních transakcí

Navíc v dnešní době, kdy jsou sklady přetvářeny jako automatizované prostory, je důležité nalézt nejvýhodnější kombinaci mezi manuálním a zautomatizovaným manipulačním systémem.

3.3 Syntaxe BPMN

Grafická notace **Business Proces Modeling Notation** (**BPMN**) je prostředkem pro modelování všech procesů, které mohou být v podnicích. Úroveň modelování není nijak omezena, takže může být notace použita i pro modelování nejzákladnějších algoritmů, které v daném podniku jsou.

3.3.1 Historie syntaxe

BPMI

O vznik konceptu **BPMN** se zasloužila nezisková skupina **BPMI**[6] [16], která se od začátku své existence (2000) snažila podporovat standardizaci obchodních procesů jako prostředek k prosazení rozvoje **B2B**. Skupina byla vytvořena 16 lídry e-businessu, kterými byli například **Black Pearl**, **Blaze Software**, **Bowstreet**, **Entricom** a další. Dalším působením se skupina rozrostla a zahrnovala více než 80 firem, které se podílely na vývoji pomocného konceptu. Jako dílčí cíle si stanovili: podporovat a rozvíjet využívání **BPM**[7] prostřednictvím vytvoření standardů pro návrh procesů, nasazení, provedení, údržbu a optimalizaci. **BPMI** chtěla, aby celý proces byl snadnější a efektivnější pro podniky na globálním trhu. V době vzniku (a myslím, že to platí i v současné době) se vyskytovaly v e-businessu naprosto nevhodné způsoby fungování procesů v rámci organizací. Např. existují různé termíny pro stejnou položku, každé oddělení používá jinak proces, který by měl být jednotný atd. Cílem bylo právě překonat tyto problémy a vytvořit standard, jak efektivněji komunikovat, sdílet data i aplikace. Samotný koncept poté obsahuje tři hlavní komponenty, a to veřejné rozhraní a dvě soukromé implementace. Veřejné rozhraní (interakce mezi dvěma obchodními partnery) je podporováno protokoly spojenými například s **BizTalk**, **ebXML** a **RosettaNet**. Další modely jsou soukromé, které jsou specifické pro každého partnera. Koncept ovšem chtěl vystavět tyto soukromé modely na všeobecném jazyce, který by definoval stejná pravidla pro každého partnera. Tak byly položeny první základy jazyka **BPML**, což je jazyk založený na syntaxi **XML**. Tak může být jazyk použit pro modelování komponent

[6] Business proces management Initiative
[7] BPM – Business Process model

podnikových dat. Přidruženým jazykem, který zastával roli dotazovatele, byl jazyk **BPQL**[8], a byl standardem pro správu. Ten bylo možné použít pro nasazení a spuštění definovaných obchodních procesů. Tyto jazyky byly otevřené specifikace a každý si je mohl stáhnout z oficiálního webu skupiny **BPMI**. Následně však bylo nutné prezentovat tyto jazyky do jednotné uchopitelné formy, které by porozuměli všichni zainteresovaní lidé.

BPML[9]

Abychom mohli pochopit, čím **BPMN** je a proč je tato syntaxe vytvořena, je dobré se podívat i na to, co tato notace představuje, tedy jazyk **BPML**. Podle oficiálního zdroje skupiny **BPMI** z roku 2002 [17] je to **Meta**, pro modelování podnikových procesů a obchodních dat. Poskytuje abstraktní model provedení spolupráce a transakčních podnikových procesů. Schopnost **BPML** je určena pro **Mission-critical** aplikace tak, že podporuje synchronní a asynchronní distribuované transakce. To nabízí spolehlivý bezpečnostní mechanismus. Sám je používán v integrovaných vývojových prostředích, řídících schopnostech projektování domů a modelů obchodních procesů přes internet[34]. **BPML** má také přidružený podnikový dotazovací jazyk na realizaci obchodních procesů[21][22].

Jiná definice z oficiálního článku [20] může být tato: *„**BPML** obecně vymezuje abstraktní model a gramatiku, které se používají k vyjádření obecného postupu.“* Jako jazyk takový, může být použit pro definování podnikových procesů, komplexních webových služeb a pluralitních prací. Základní díly, které tvoří **BPML** jsou abstraktní **BPML** konstrukty syntaxe **XML** a specifikace, které v **BPML** jsou **Namespace**, **Features**, **Imports**, **Target Namespace**. Aktivity v **BPML** vykonávají specifické funkce a jsou bud jednoduché či složité. Jednoduché úkony jakou jsou akce, volání, kompenzování atd. Nelze je dále rozložit a provést jedinou operaci.

[8] BPQL – Business Process Query Language- univerzální procesní dotazovací jazyk pro modelování podnikatelských procesů

[9] BPML Business Process Modeling language – jazyk pro modelování podnikatelských procesů, jazyk založen na bázi XML.

Komplexní aktivity jsou všechny sekvence, přepínače a atd. Jsou složeny z jedné či více aktivit a řídí vykonávání určité aktivity pomocí setu aktivit.

První zmínky a myšlenky BPMN

Notací, kterou **BPMI** vytvořila pro grafické znázornění **BPML**, byla **BPMN**, a jak vyplývá z materiálu vydaného skupinou **BPMI** z roku 2002 [17], pracovního návrhu 0.9 cílem tohoto konceptu je: Poskytnout notaci, která bude snadná a pochopitelná pro všechny firemní uživatele, od obchodního analytika, který vytváří počáteční návrh procesu, až po technického vývojáře zodpovědného za implementaci technologie, která bude provádět tyto procesy. Takže **BPMN** vytváří most, který překlene propast mezi procesní analýzou a implementací procesů. Někdy je ovšem problém s časovým vymezením procesu[19]. Dalším cílem je, aby jazyky na bázi **XML**, (**BPEL4WS**[10] a **BMPL**[17][21][22]) bylo možné formulovat po vizuální stránce za pomoci jednoho standardu.

Dále je v dokumentu [17] **BPMN** uvedeno, že se koncept bude modelovat do **BPD** (**Business Process Diagram**), kde budou jednotlivé prvky (objekty) propojeny a budou vytvářeny pomocí vztahů sekvenčních, paralelních či alternativních cest. Koncept **BPMN** musí být ovšem natolik jasně a srozumitelně podaný, aby jej pochopili všichni uživatelé, ale i natolik složitý, aby mohl obsáhnout všechny možné procesy, které budou dnes či v budoucnu modelovány.

Workflow s

V tomto dokumentu [24] jsou už ucelené formulace jednotlivých elementů, které se používají v syntaxi **BPMN**. V této fázi vývoje byly vytvořeny základní elementy, kde se ještě nebraly v potaz přerušované a nepřerušované události a některé další typy událostí. Taktéž nebyly brány v potaz jednotlivé druhy **Tasků** (**Manual**, **Servis** atd.). To vše je pochopitelné, jelikož dokument je směřován na verzi **BPMN** 0.9[17] a některé věci, které jsou dnes hojně využívány, vznikly až ve verzi **BPMN** 2.0[26] Ovšem už v této verzi jsou k vidění první verze **Design Patterns**, kterých sice bylo v dokumentu na začátku asi okolo 10, nyní je jich definováno 43 [41]. I tak lze předpokládat, že už od začátku skupina chtěla, aby se

[10] BPEL4WS (Business proces Exectuion Language for Web Services Jazyk založen na formátu XML sloužící pro grafické znázornění workflow webové služby. Dnes je znám pod WS BPEL

používala syntaxe podle určitých standardů, a aby se modelovalo podle stejného či podobného stylu. Nevytvářela tedy úplně nový styl myšlení, ale chtěla využít dostupné předpisy a definovat, jak modelovat a vytvářet něco nového. Proto zpracovala jen grafický koncept pro vymodelování, údržbu a inovaci podnikových procesů, pro které byla už definice položena v minulých letech před vznikem skupiny **BPMI**. I **Workflow Patterns** byly už ovšem definovány v 90 letech 20. století, kdy je popsali i zainteresovaní akademici z Univerzity v Eindhovenu, jak dokládá dokument „**Advanced Workflow Patterns**"[24]. V jiném dokumentu z roku 2002 od téže univerzity s názvem „**Worflow Patterns**"[23] je popsáno do detailu 20 **Workflow Patternů**, které jsou dodnes stále používány.

Myšlenka o BPMN

Je zajímavé číst si z původního materiálu této skupiny. Nejenom že zde přidává svůj názor na současnou situaci v procesech a notaci **BPMN**, ale snaží se i trochu předpovědět, jakým směrem se bude tato syntaxe směřovat.

Primárním jazykem pro tuto notaci byly jazyky **BPEL4WS** a **BPML** [21,22,23], a tak se bude **BPMN** vztahovat pouze k těmto jazykům. Mapování jiné specifikace si každý bude muset nadefinovat sám. I zde říkají [23], že je těžké předpovědět, které mapování bude pomocí **BPMN** preferováno, protože specifikace jazyka jsou volatilní oblast. Další podstatnou informací, kterou by si každý „systémový architekt" měl uvědomit, že **BPD** není navržen tak, aby graficky předával veškeré požadované informace o spuštění podnikových procesů. (Dle mého názoru se nepopisuje celý systém úplně do detailu, ale modeluje se do takové úrovně abstrakce, aby jí mohl každý uživatel porozumět). Přemrštěná detailnost při modelování vede k chaosu a jeho prvotní popsání ztrácí smysl.

Jaké máme verze a kdy přišly verze předchozí[25].

BPMN 1. X

Květen 2004, byla představena verze 1.0

V lednu 2008 po sloučení se skupinou **OMG**[11] v roce 2006 vyšla verze 1.1[17], která upřesňovala některé chyby, které byly ve verzi 1.0[17].

[11] OMG - Object management Group

Rok na to byla vydána verze 1.2[43], která byla pouze dovětkem k verzi 1.2[17]. Byly zde upraveny pouze menší chyby, které se nestihly vytvořit v předchozí verzi a bylo nutné je upravit.

BPMN 2. X

Leden 2011 byla vytvořena verze 2.0[26], která měnila koncept předchozích verzí. Tato verze je také považována za přelom. O tento koncept byl zvýšený zájem pro jeho rozmanité vyjádření situací. Dosavadním lídrem tohoto konceptu byl spíše aktivity diagram v syntaxi **UML**. I když někteří autoři si myslí, že koncept **UML** s aktivity diagramem je lepší než **BPMN**, je tomu spíše naopak. Na toto téma byly prezentovány mnohé myšlenky a články, které s tímto závěrem nesouhlasí. Na akademické půdě byla vytvořena i celá řada diplomových prací, které právě řeší rozdíl mezi **UML** aktivity diagramem a **BPMN**. Například zde je možné si přečíst, že jsou tyto syntaxe vyrovnané, avšak některé procesy nejdou svou komplexností znázornit pomocí **UML** .

Poslední verzí, která byla zatím prezentována pro veřejnost, byla z roku 2012 verze 2.0.2 [27]

A co bude přesně tedy BPMN zobrazovat?

BPMN bude zobrazovat procesy jednotlivých účastníků, z nichž každý účastník může zobrazit schéma rozdílně. To znamená, že účastníci mají různé úhly pohledu na to, jak se tyto procesy budou chovat. Tyto procesy mohou být pro účastníka interním procesem, tedy procesem, který se vztahuje k účastníkovi, nebo tyto procesy budou externí, kdy zde bude účastník figurovat jako pouhý uživatel. Každý účastník tedy bude mít jinou perspektivu, jak se bude na všechny procesy dívat, pokud jde o vnitřní a vnější procesy. Jde tedy o jiný úhel pohledu účastníka, avšak sám diagram, model či proces zůstávají stejné. I když sám diagram je velmi důležitý pro každého diváka, aby mohl divák porozumět procesu, **BPMN** nebude v současné době (verze 0.9)[3] podporovat definování jakéhokoliv mechanismu, který by určil, o jaký pohled se jedná. V současné době se **BPMN** už odlišuje v pohledu na procesy[26,27]. **BPMN** je rozděleno do několika druhů diagramů, který každý sám za sebe představuje jiný pohled na procesy. Koncept je otevřen pro modeláře, ale také může být využit (a je to doporučeno) jako vhodný pomocník

pro prodejce, kteří mohou své procesy vymodelovat do vizuální stránky a tak i lépe nastínit a charakterizovat své procesy zákazníkům.

V dalších částech dokumentu už jsou rozepsány jednotlivé elementy tak, jak se používaly. Je zajímavé, že popisy, které jsou v tomto dokumentu obsaženy, se stále používají a byly změněny buď pouze v malém měřítku, nebo vůbec. Takže i po sloučení skupiny **BPMI** se skupinou **OMG** se používala podobná terminologie, která byla před tím.

Jako dobrý zdroj lze doporučit prezentaci od Stephen A. White, která popisuje minulost, přítomnost a budoucnost konceptu **BPMN**. Tato prezentace je z roku 2012 a bude přílohou[25].

Minulost

Na začátku je taktéž vymezen záměr, který zakladatelé měli. A to:

BPMN poskytne podnikům schopnosti porozumět jejich vnitřním a vnějším obchodním postupům s grafickou notací a dá organizacím schopnost komunikovat pomocí této syntaxe s ostatními.

A i když mnozí uživatelé tento koncept využívali, mnoho z nich bylo nespokojeno s verzí, protože některé postupy nebylo možné vymodelovat a komplexnější řešení nebylo možné ztvárnit.

Současnost (2012)

Několik let od vzniku (10 let) se už využívá syntaxe **BPMN** jako standard, bez kterého není možné existovat. V roce 2012 bylo 45 firem, které používaly ještě **BPMN** verzi 1. X, a už 25 firem přešlo plně na **BPMN** 2.0[25]. I když tyto informace jsou podložené pouze prezentací, tento zmíněný autor S. White vydal mnoho publikací, které jsou hodnotné a velmi ceněné. Jak je tomu dnes, v roce 2017, není zatím známé, ale díky otevřenosti syntaxe je už řádově několik tisíc koncových uživatelů.

Pokud bychom měli definovat, jestli byl jejich cíl zatím splněn, musíme si říct, že se **BPMN** rozmohl a byly vytvořeny další diagramy, které více prohlubují jejich význam pro společnosti. Interní procesy se staly více kontrolovanými normativními typy a externí procesy jsou namodelovány prostřednictvím právě nově vzniklých diagramů (2012) **Collaboration, Conversation a Choreography**[27]. Bohužel v současné době také záleží na typu nástroje, jakým

modelujeme. Nemůžeme modelovat pomocí nástroje, který plně nepodporuje syntaxi **BPMN**, nebo který ji částečně mění pro svoje využití. I když tato syntaxe je volná, měla by platit základní pravidla, která se budou dodržovat. Avšak i tak můžeme říct, že se **BPMN** vyvinulo v takový koncept, který se používá i pro modelování organizačních procesů. Tedy manažer více využívá tuto notaci, a když nastane krizová situace, nebo když se hledá viník za případné selhání, pomocí vymodelování procesů například ve výrobě ví, jaký člověk či jaká činnost mohla způsobit vzniklý problém.

Jako další jsou popsány nové prvky, které byly přidány do syntaxe ve verzi 2.0. Například **Ad-hoc**, který je [27] zde popsán jako špatně pojmenovaný element.

V dokumentu[25] se také můžeme dočíst, že samotná notace **BPMN** způsobila na trhu mnoho změn. Koncoví uživatelé si uvědomili potenciál této syntaxe, což dokládá fakt, že jen v roce 2012 bylo 70 nezávislých nástrojů[25], které podporovaly tuto syntaxi a vytvořily tak vhodné prostředí pro modelování procesů pro tento koncept. Ovšem dokládá, že se syntaxe bude neustále vyvíjet, jako vše ostatní. I když je verze 2.0.2 [27] už dobře vybavena a poskytuje pro uživatele dobré prostředí, stále je mnoho míst, kde je nutné zapojit fantazii a trochu zneužít pravidla notace, aby mohla být situace zmapována. Takže se ohýbá syntaxe a není používána naprosto správně. I když je notace volná a naše fantazie je nutná (říká se, co analytik, to jiné myšlení) platí, že jeden proces může být zmapován různě, ale hlavně musí být správně.

Co ale přinesou další léta pro syntaxi **BPMN**? Dnes 2017 máme i **ISO** předpis pro modelování **ISO/IEC 19510:2013**[44], takže se máme držet určitých pravidel, aby vše bylo jasně srozumitelné a nevyskytovaly se nejasnosti. Notace se sama bude určitě vyvíjet, ale abychom ji i my koncoví uživatelé mohli dobře uchopit a správně modelovat, bude zapotřebí, abychom se i my neustále vyvíjeli. Proto je nutné vytvářet sofistikovanější metodiky, jak modelovat, což v dnešním světě (v České republice je to k vidění), je velice nutné. Každá firma si modeluje podle sebe, ale co skutečně potřebujeme jako uživatelé, je jasná a jednotná metodika. Bohužel firmy samy o sobě to nikdy nezveřejní, protože je to jejich know-how, a takové znalosti jsou ceněny zlatem. Tady vidím možnost vysokých škol, které v rámci dotací mohou být nositeli těchto metodik. V minulosti se to

osvědčilo například s **Workflow Patterns**. Taková metodika by byla ceněna a zakladatel uznáván.

3.3.2 Elementy syntaxe BPMN 2.0.2

V BPMN rozlišujeme několik typů elementů, se kterými se můžeme při modelování setkat. Tyto prvky podle určitých standardů na sebe navazují a podle této návaznosti je čten celý diagram. V základním rozdělení určujeme 4 skupiny elementů. Jsou to Tokové objekty (**Flow Objects**), Spojovací objekty (**Connecting Objects**) **Pool, Lany** a Artefakty (**Artifacts**)[26].

Tokové objekty

Tyto prvky jsou základními grafickými prvky, které mohu vytvářet a tak i definovat celý proces. Tato skupina sama osobě může být použita pro tvorbu celého procesu. Ostatní skupiny jsou určeny pro další doplnění samotného procesu přidruženými elementy. Jako tokové objekty můžeme definovat: Události, Aktivity a Brány (**Gateway**). Dále můžeme rozlišovat tyto jednotlivé elementy.

Události

Tento typ elementu znázorňuje děj, který zahajuje, upřesňuje či ukončuje celý proces. Události můžeme dělit na události, které přijímají a události, které odesílají. Přijímající události, jsou znázorněny jako obrysy určitého typu události a jsou taktéž nazývány v anglické literatuře jako „**Catching**“. Odesílající událost je znázorněna jako černý znak určitého typu události a v anglické literatuře je nazvána jako „**Throwing**“[26]. Dalším rozdělením událostí je pohled na přerušení či nepřerušení vyvolané události. Přerušovací události, které mají schopnost přerušit danou aktivitu, se nazývají **Interrupting**. Modelují se převážně na začátku procesu, nebo v případě, kdy má být proces předčasně ukončen či přerušen. Druhým typem událostí, které mají vliv na proces, jsou nepřerušující. Tyto události se mohou modelovat do procesu a jejich vyvolání nezpůsobí ukončení procesu, ale vytvoří se druhý tok procesu, který běží s původním tokem procesu paralelně. Tedy tato událost je nepřerušující, v anglicky psané literatuře **Non-Interrupting**. Události s vlastnostmi **Non-interrupting** se značí jako obvyklé události, ovšem

s přerušovaným obvodovým krajem. Tyto události jsou poté rozlišovány dalšími specifikacemi[26].

Typy událostí

Message – Spouštěč, který představuje událost, která zahájí proces díky přijetí zprávy. Tato zpráva může být ve formě elektronické či fyzicky předaná příjemci. **Event** typu **Message** může být, jak je patrné z tabulky, typu **Start**, **Intermediate** i **End**. Každý typ má podobnou funkci. U **Intermediate Eventu Message** může mít dva významy. Když je **Intermediate Event Message Catch**, znamená to, že daná událost je přijetí zprávy a **Intermediate Event**. **Message Throw** je událost, která znamená odeslání zprávy příjemci, po odeslání se pokračuje v toku. Určení směru zprávy může být určeno pomocí vazby **Message flow**, která bude popsána níže[26].

Timer – Událost definovaná jako určitý časový okamžik. Tento časovač udává, v jakou dobu se daný proces spustí. Může to být měsíční kontrola na skladě, či týdenní kontrola materiálu ve výrobě. Taktéž může naznačovat, když je **Event** na hranici aktivity čas, ve kterém je možné danou aktivitu vypracovat. Pokud čas od spuštění aktivity překročí daný interval, aktivita je ukončena a pokračuje tokem dál[26].

Conditional – Událost, která je spuštěna za určitých podmínek. Například při snížení pod minimální hranici zásoby produktu nebo vzniku krizové události. Tento **Event** může být typu **Start** a **Intermediate**, protože slouží ke spuštění určitého procesu či pokračování daného procesu[26].

Error – Událost, která je spuštěna vzniklou chybou. Může být typu **Start**, **Intermediate** či **End**. Při **Intermediate** může být **Event** pouze připojen na hranici aktivity a značí její chybu, která může v dané aktivitě nastat. Při vzniku chyby poté vede tok **Event** dalším směrem podle směru vazeb[26].

Compensation – Událost používající se v transakčních aktivitách k modelování požadavku na „**Undo**“. Událost zajišťující návrat do předchozí aktivity v případě, že byla transakce zrušena[26].

Escalation – Událost, která značí eskalaci na vyšší úroveň rozhodování. Používá se v **sub-procesech**. Příkladem může být činnost pracovníka, který při dané práci

potřebuje potvrzení od svého nadřízeného. Tento typ **Eventu** je někdy popisován jako nadbytečný **Event** v syntaxi[26].

Cancel – událost využívající se v **sub-procesech**, kdy je daný proces ukončen.

Link – Událost zpřehledňující model procesu. Využívá se při křížení **Sequence flow**, daný tok je rozdělen na dvě části. Na jedné straně vchází tok do **Catching** události typu **Link**, za potencionálním křížením se použije **Throwing** událost typu link, ze které vychází **Sequence flow** do požadovaného elementu[26].

Signal – Událost tohoto typu je spuštěna při přijetí určitého signálu, který může vzniknout na jiném místě systému. Tento **Event** je **Start Event, Intermediate Event** a **End Event**. Při **Start Eventu Signal** se vždy počítá s přijetím signálu a při endu, jako vznik signálu pro spuštění jiného procesu. **Intermediate Event** může být jak **Catch** tak i **Throw**, jelikož v průběhu procesu může vzniknout situace, kdy vznikne signál pro jiný proces, nebo právě v průběhu procesu může být situace, kdy je nutné přijmout signál, aby mohl daný proces pokračovat[26].

Terminante – Událost, která ukončuje všechny aktivity a celý proces. Tento druh události je ve variantě pouze s ukončující aktivitou[26].

Multiple – Spouštěč definující několik způsobů, jak daný proces může začít. Na aktivaci události je však potřeba pouze jedna z podmínek. Dané podmínky mohou být slovně či matematicky popsány[26].

Parallel Multiple - Událost podobná události **Multiple**. S tím rozdílem, že na zahájení pokračování toku musí přijít všechny podmínky, aby byl tok spuštěn. Tedy všechny podmínky musí být splněny[26].

Conditional – Událost, která může spustit nebo umožnit pokračování daného procesu pouze za předpokladu, že je podmínka splněna. Do doby než je podmínka splněna, tok je ve stavu vyčkávacím. Až bude daná podmínka splněna, tok se bude provádět dál[26].

Rozdělení událostí podle umístění v diagramu

Start Event - Událost, která stojí na začátku procesu i **sub-procesu**. Značí počáteční událost, která zahájí tok daného procesu. Tento typ **Eventu** může být vždy typu **Catch**. Jelikož celý proces začíná touto událostí, nemůže být hned na začátku procesu, aby startovací událost odesílala zprávu jinému procesu [26].

Intermediate Event – Událost, která je v toku procesu, je to takzvaný mezi-vstup určující například další pokračování toku, spolu s jedním typem **Gatewaye** slouží jako rozhodovací určení směru toku procesu. Dále může sloužit jako událost, ve které se odesílá zpráva jinému zainteresovanému účastníkovi daného procesu či jako přijímač zprávy od jiného účastníka. Dalším možným principem této události je taktéž časové určení. Tedy daný proces bude pokračovat či jiné[26].

End Event – Ukončující událost. Tato událost stojí vždy na konci procesu. V jednom procesu může být několik ukončovacích aktivit. Jedna ukončující událost může být začáteční událostí jiného procesu. Tento typ **Eventu** může být pouze typu **Throw**. Jelikož za touto událostí neleží žádná aktivita a je proces touto událostí ukončen, může pouze posílat zprávu, signál.

Obrázek z přílohy č. 1. ukazuje celkové rozdělení událostí podle typu události. Je patrné, že ne všechny kombinace typů událostí jsou možné. Například událost **Link** je pouze typu **Intermediate**. Takto je zřejmé, jaké kombinace daného typu události jsou k dispozici pro modelování procesů. Taktéž lze ze sloupce **Boundary** u **Intermediate Eventu** vyvodit, jaké elementy se mohou připojit k aktivitě na její hranici. Toto znázornění je na obrázku č. 1. Tyto **Eventy** označují například nečekané pokračování aktivity. Celkový počet událostí, se kterým modelář bude pracovat, je 63 typů. Podle některých tvrzení vede ovšem takové množství událostí pouze ke zvýšení chybovosti při modelování. Některé typy událostí jsou považovány za špatně popsané a zbytečné[26].

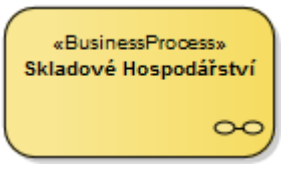
Obr. 1 Aktivita s Eventem
Zdroj: vlastní tvorba

Aktivity

Aktivity jsou činnosti, které vyjadřují v procesu určitou činnost. Tyto činnosti jsou na sebe navázány a vytváří proces. Aktivity mohou být několika typů, které se při modelování odlišují svým významem[26].

Business Process

Business Process slouží jako základní rozdělení celého systému na jednotlivé procesy. Tyto procesy jsou

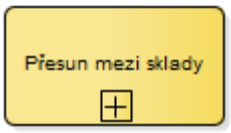


Obr. 2 Business Process
Zdroj: vlastní tvorba

dále rozděleny na **sub-procesy** či atomické **Tasky**. **Business Process** tedy leží na nejvyšším hierarchickém místě. Je označován jako obdélník s oblými rohy s malým procesem v pravém dolním rohu.

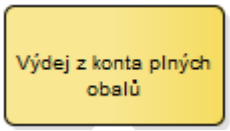
Aktivita

Činnost, která probíhá v rámci procesu či **sub-procesu**. Jsou dva druhy aktivity, které se v diagramech modelují a liší se rozsahem rozpadu. Název aktivity musí být vypovídající. Proto je nutné psát danou aktivitu jako probíhající činnost. Například vyskladnění nebo vytvoření prodejní objednávky[26].



Obr. 3 Sub-process Zdroj: vlastní tvorba

- **Sub-proces** – Aktivita, která je charakterizována dalším rozpracováním dílčích činností, které obsahuje. Vzniká tak v aktivitě další proces podrobnějšího vysvětlení dílčího stavu. Takto se může členit celý systém do několika úrovní.
- **Task** – Aktivita, která není dál rozepisována do podrobností a nevytváří další dílčí proces. Zůstává nedělitelná.



Obr. 4 Task Zdroj: vlastní tvorba

Vytvořené aktivity jsou v defaultním stavu brány jako **Tasky** a příslušnou změnou ve vlastnostech aktivity změní daný typ na **Sub-proces**. Při modelování aktivit v procesu může taktéž modelář využít typu aktivit a 3 značky pro jejich větší specifikaci. Značky **Multi-instance** nebo **Loop** můžou být použity spolu s **Compensation.** Značky **Loop** a **Multi-instance** nemohou být vytvářeny společně.

Multi-instance

Obr. 6 Multiinstace 1 Zdroj: vlastní tvorba

Multi-instance je značka pro vytváření mnoha instancí jedné aktivity. Tedy když vytváříme vychystávací příkaz, máme x produktů k vyskladnění a musíme vytvořit na každý druh produktu jeden řádek, kde budeme specifikovat množství a místo uskladnění. Proto můžeme zavést **Multi-instance**. U každého Tasku **Multi-instance** můžeme ještě specifikovat, jestli budou prováděny jednotlivé instance paralelně či sekvenčně. Paralelně prováděná **Multi-**

Obr. 5 Muliinstance 2 Zdroj: vlastní tvorba

instance má značku tří svislých čar vedle sebe a sekvenční značka **multi-instance** je znázorněna jako tři vodorovné čáry nad sebou. V tomto případě by to byla sekvenční značka.

Loop

Obr. 7 Loop Zdroj: vlastní tvorba

Jedná se o opakování celé činnosti, podle zadaných podmínek se provádí opakování celé aktivity podle stavu podmínky. Pokud podmínka už není splněna, tok pokračuje dále. Je to například při plnění obalových jednotek na skladové umístění. Pracovník naplňuje dané umístění do doby, než je zaplněné. Poté už není možná na plné umístění něco umisťovat a svou činnost končí. Značí se šipkou, která ukazuje na svůj konec. U daného opakování nemusí být znám počet opakování[26].

Compensation

Obr. 8 Compensation Task Zdroj: vlastní tvorba

Tato značka je v případě, kdy musíme danou aktivitu určitým způsobem změnit při nějaké podmínce či v daném okamžiku. Je to aktivita, která není zařazena do běžného toku procesu, ale je ponechána stranou vedle aktivity, kterou „kompenzuje“. Tato kompenzující aktivita je bez toku. Například vytváříme rezervaci určitého produktu na skladě pro jednoho spotřebitele. Ovšem může nastat okamžik, kdy daný spotřebitel například prodloužil dobu dodání z týdne na 2 měsíce. Daný produkt by byl zbytečně připravován pro jednoho spotřebitele a tak dojde odrezervování. Značí se dvěma šipkami vedle sebe ve spodní polovině aktivity[26].

Typy tasku

Obr. 9 Service task Zdroj: vlastní tvorba

Service Task – Service Task slouží pro znázornění tasku, který je vykonáván určitou webovou úlohou nebo je automaticky vytvářen systémem. Do této aktivity tedy nevstupuje aktér, ale je vytvořena zcela bez pomoci lidského prvku.

Obr. 10 Send task Zdroj: vlastní tvorba

Send Task - Send Task je aktivitou, ve které je vytvořena a odeslána zpráva pro určitého aktéra. Při odeslání je aktivita ukončena a pokračuje se v toku procesu.

Receive Task - **Receive Task** je obdobný, jako **Send Task** pouze s tím rozdílem, že daná aktivita čeká na příjem určité zprávy od aktéra a poté je ukončena a pokračuje tok procesu. Může nastat situace, kdy je v jednom procesu použit **Send Task** který odesílá zprávu na **Receive Task** jiného procesu. Jedná se tedy o odeslání a přijetí zprávy v jednom systému.

Obr. 11 Receive Task
Zdroj: vlastní tvorba

User Task – Nejběžnější typ **Tasku**, který se vyskytuje ve všech systémech. Je to aktivita, kde pro vytváření dané činnosti je zapotřebí lidský prvek, který spolu se softwarem vykonává danou činnost. Je to například aktivita Vytvoření výdejního listu.

Obr. 12 User task
Zdroj: vlastní tvorba

Manual Task – Aktivita, kterou vykonává aktér sám bez potřeby zapojení softwaru do dané činnosti. Je to například manuální zaskladnění produktu na umístění.

Obr. 13 Manual task
Zdroj: vlastní tvorba

Business rule – Aktivita, která je vytvářena pod záštitou Business pravidel. [26]

Obr. 14 Business rule task
Zdroj: vlastní tvorba

Script Task – Aktivita bude vytvořena v určitém jazyce **Scriptu**, který je definován modelářem či implementátorem. Při dokončení **Scriptu** je daná aktivita taktéž ukončena.

Obr. 15 Script task
Zdroj: vlastní tvorba

Gateway

Poslední skupinou v tokových objektech jsou brány. Tyto elementy slouží pro upřesnění cesty. Například za určitých podmínek půjde tok jednou cestou, za jiných podmínek půjde tok druhou cestou. Tyto elementy nám pomáhají v upřesnění a řízení toku. Jelikož je mnoho různých způsobů a případů, jak tok vést, existuje i několik **Gateway** elementů[26].

Exclusive Gateway

Obr. 16 Exclusive Gateway Zdroj: vlastní tvorba

Tento element je pro první pochopení velmi jednoduchou branou. Tok bude veden tou cestou, pro kterou má splněné podmínky. Může tedy jít bud cestou A, nebo B nebo C. Přes tento **Gateway** nemůže jít více cestami najednou ale pouze jednou, která splňuje podmínku. Pokud tok nesplňuje žádnou podmínku, může pokračovat z brány pomocí **Sequence flow** typu default. S **Exclusive Gateway** se taktéž pojí terminologie hlavní a vedlejší cesta. Ta cesta, jež je vybrána, je hlavní a ostatní nespuštěné jsou vedlejší cesty. Všechny cesty jsou vzájemně výlučné. Tato vazba bude popsána níže. **Gateway Exclusive** má jeden problém, a to otevřenost elementu. Nad touto otázkou se vede mnoho diskuzí na vědecké úrovni a není lehké na ni jednoznačně odpovědět[26].

Inclusive Gateway

Obr. 17 Inclusive Gateway Zdroj: vlastní tvorba

Tento element má podobný význam jako **Exklusive Gateway** s tím rozdílem, že po vyhodnocení podmínek mohou být spuštěny všechny cesty. Vše závisí na tom, jaké podmínky jsou nastaveny a jestli je dané toky splňují. Když mluvíme o tocích z **Inclusive Gateway**, mluvíme o nezávislých tocích[26].

Parallel Gateway

Obr. 18 Parallel gateway Zdroj: vlastní tvorba

Tato brána rozděluje jeden tok na několik paralelních vláken. Jelikož ovšem **BPMN** nepopisuje časové rozložení procesu, může se stát, že tyto cesty neproběhnou paralelně v jednom čase, ale po sobě. Je ovšem důležité, v jaké části fázi procesu se tok rozděluje, proto se modeluje takovým způsobem[26].

Event-Based Gateway

Obr. 19 Event-based gateway Zdroj: vlastní tvorba

Tato brána zastavuje tok a určuje směr potencionálního toku. Směr, který bude v procesu určen, jako hlavní závisí na **Eventu**, který spustí jednu či druhou cestu. Jakmile je jedna událost

aktivována, daný proces se ubírá tokem, na kterém je spuštěná událost. Druhá, nyní vedlejší cesta, je uzavřena a nemůže být spuštěna, i kdyby byla spuštěna událost na ni přidružená[26].

Parallel Event-based Gateway

Obr. 20 Parallel event-based gateway Zdroj: vlastní tvorba

Tato **Gateway** má vlastnosti **Paralell Gateway**e a **Event-Based Gateway**e. V dané bráně je rozdělen daný tok na několik cest, které jsou podmíněny aktivací události, jež jsou na daných tocích.

Complex Gateway

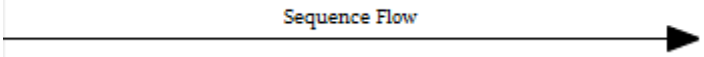
Obr. 21 Complex gateway Zdroj: vlastní tvorba

Complexy Gateway je modelován do komplexních diagramů, kde je zapotřebí například složitá synchronizace toků či jejich rozdělení. Základem je snaha modelovat vše pomocí jednodušších **Gateway**, které byly popsány výše. Ovšem jsou situace, kdy bychom museli na vymodelování použít i několik **Gateway** za sebou. Právě pro tyto komplexní situace je tento **Gateway** vytvořen. Jelikož podmínka není jednoduchá, musí být dopodrobna popsána v daném **Gateway**.

Spojovací objekty

Objekty, které slouží pro názorné členění posloupnosti jednotlivých elementů či toku procesů a komunikace uvnitř procesů [26].

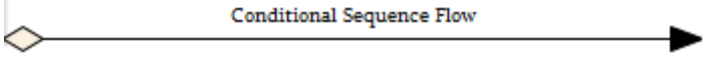
Sequence Flow



Obr. 22 Sekvenční tok Zdroj: vlastní tvorba

Sekvenční tok, který spojuje jednotlivé elementy mezi sebou v rámci toku. Sekvenční tok nám udává směr toku procesu, kterým se ubírá.

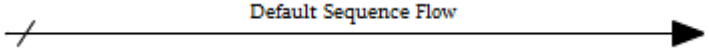
Conditional Sequence Flow



Obr. 23 Conditional sekvenční tok Zdroj: vlastní tvorba

Conditional sequence flow je vazba, která je vedena od elementu k elementu v závislosti na podmínce. Tedy tok se může ubírat touto cestou, pouze pokud splňuje podmínku.

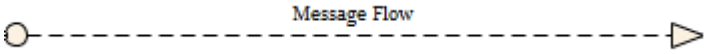
Default Sequence flow



Obr. 24 Default sekvenční tok Zdroj: vlastní tvorba

Vazba, která je aktivována, pokud není v daném elementu podmínka nebo nejsou žádné směry,

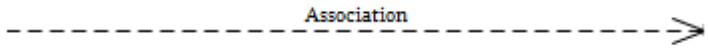
kterými se může tok dát vést. Tato vazba se může vést z aktivity, **Gateway** i z události.



Obr. 25 Message flow Zdroj: vlastní tvorba

Message Flow

Vazba sloužící ke komunikaci mezi pooly. Jedná se o zprávy, které jsou zasílány od dodavatele či jiného systému k odběrateli. Tyto zprávy posílané mezi těmito partnery mohou sloužit jako spouštěč pro další procesy.



Obr. 26 Asociace Zdroj: vlastní tvorba

Association

Vazba asociace je obecný vztah mezi elementy a artefakty. Může naznačovat u datových objektů, zdali jsou vstupující či vystupující.

Pool, Line and Artefacts

Tyto objekty slouží pro proces jako doplňující informace, které daný proces více zpřesňují tak, aby byl pro čtenáře jasnější.

Pool and Lane

Objekty rozdělující proces v rámci jednoho systému do dílčích skupin.

Pool

Objekt **Pool** rozděluje proces na jednotlivé oblasti, ve kterých je daný proces tvořen. Když máme systém komunikující s druhým systémem, dáme procesy spojené s jedním systémem do jednoho poolu a jiné procesy, které se tvoří v rámci jiného systému do dalšího poolu. Mezi **Pooly** neexistuje spojení jiné než přes **Message flow**. Toto pravidlo je víceméně jasné. Pool představuje jeden systém, tedy jedna pravidla, jednotné tvoření procesů v rámci oblasti. Je to otázka dat. V rámci firmy můžeme mít tento příklad. Firma má několik vnitřních systémů. Na výrobu, na sklady a expedici, na nákup. Tyto systémy jsou různé a nemají stejné jádro. Data se musí přesouvat z jednoho systému do druhého pomocí ručního přepisování nebo generování. Zde je jasné, že všechny systémy v rámci jedné firmy jsou pooly a musí být takto brány. Ovšem když máme firmu, kde je například **ERP** systém tvořen na jedné platformě, jedná se o jeden systém, a to i když se na vývoji jednoho systému podílí několik firem. Základna systému je jednotná a tudíž i

oblasti v ní. Skladové hospodářství ekonomika, personalistika, **CRM** jsou brány, jako vnitřní dělení jednoho systému, tedy **Lane** [26].

Zhroucený pool

Je to bazén, který je modelován jako pohled na komunikaci mezi pooly, nejsou známy procesy, které probíhají uvnitř.

Otevřený pool

Pool, ve kterém jsou známy procesy, které daný systém tvoří. Je jasně definované místo, odkud se komunikuje s jiným poolem a místo, kde se přijímají zprávy od jiného poolu.

Lane

Tyto objekty jsou dílčím rozdělením poolu. Tedy jak bylo zmíněno výše, v rámci jednoho systému můžeme dělit jednotlivé oblasti, které se podílejí na vyhotovení jednoho procesu. Nedochází k přesunu dat pomocí zpráv, ale pouze k přesunu dat v rámci jednoho systému jako celku[26].

Artefakty

Objekty rozšiřující proces o další popisy spojené s jednotlivými aktivitami či elementy daného procesu.

Data Objects

Vstupující a vystupující datové objekty, tedy datové soubory vzniklé v dané aktivitě, nebo objekty, které jsou potřebné pro vyhotovení dané aktivity. Ve skladovém hospodářství jsou to například dodací listy, které jsou důležité pro expedici a kontrolu obsahu zabalené objednávky se skutečným vyskladněnými produkty ze skladu [26].

Group

Objekt používající k rozdělení elementů podle určitých vlastností či jiného uskupení. **Group** nemůže nahrazovat **Pool** ani **Lane**.

Annotation

Objekt doplňující element o další popis, který může blíže čtenáře seznámit s elementy nebo s elementy v procesu.

Data Store

Datové úložiště ukládající nebo získávající data, které jsou v procesu potřebné pro jeho další průběh.

Message

Objekt ukazuje posílání zpráv mezi aktivitami či **Pooly.** Zpráva může sloužit jako začátek sub-procesu nebo k dokončení či startu nějaké aktivity.

Workflow Patterns

Workflow Patterns jsou ustálené standardy, které jsou doporučením při modelování určitých situací. Jak bylo zmíněno výše, tyto **Patterny** jsou již známy od 90 let 20 století a první zmínky o nich jsou obsaženy v publikaci (van der Aalst a spol., 2002). Zde jsou poprvé zmíněny **Workflow Patterny**, které se ve většině používají až do dneška. V současné době je definováno 43 **Patternů**, které jsou specializované na danou situaci a popisují, jak má modelář modelovat. **Workflow Patterny** jsou velkou součástí syntaxe **BPMN** a je vhodné při modelování tyto metody používat, jelikož také slouží jako nepsaná metodika pro modelování. Vzhledem k účelu této práce zde ovšem nejsou popsány a vymodelovány.

3.4 Syntaxe UML – Use Case

UML (Unified Modeling Language)

Jazyk pro grafické ztvárnění systému. První myšlenky o sjednoceném jazyku, podle kterého by se modeloval systém, byly už v roce 1966. Avšak až v roce 1997 byla vydána první ucelená verze syntaxe **UML**. V tomto roce taktéž syntaxi odkoupila společnost **OMG,** která rozšířila daný jazyk dalším směrem. Jazyk **UML** se používá pro analýzu systému za pomoci objektového přístupu, ale také pro detailní popis návrh systému. Syntaxe nabízí i pohled do business procesu, který je ovšem víceméně nahrazen právě výše zmíněnou syntaxí **BPMN**, kterou vlastní tatáž společnost. V této syntaxi nabízí společnost 13 diagramů pro popis systému, krom zmíněného **Use case** modelu, nabízí **Requirement** diagram, **State** diagram, **Sequence** diagram a mnoho dalších.

Use case

Model **Use Case** slouží pro zachycení komunikace uživatele se systémem. Vzájemná interakce je použita při základním pohledu na systém, rozvržení tříd a jejich komunikace mezi sebou. V některých případech se můžeme setkat s názory (Kraval I.), že je **Use case** nejdůležitější model. **Use case** navazuje na **Business Proces**s v tom, že nám upřesňuje ty aktivity, které jsou vytvářeny uživatelem za pomoci informačního systému. Právě tato komunikace je velmi důležitá, jelikož nám rozšiřuje požadavky na systém z pohledu užití a rozčleňuje hranice systému. Tento model vychází ze syntaxe **UML**[7].

Diagram je tvořen několika elementy, které plně obsahují komunikaci uživatele se systémem: Aktér, **Use case** (typová úloha), vazba, **Boundary** (Hranice systému)

Aktér

Uživatel, který interaguje se systémem, tedy vytváří akci a následně přijímá reakci systému. Osoba aktéra zde představuje skupinu lidí se stejnými pravomocemi, které mají při přístupu do systému, například Administrátor, či zákazník.

Obr. 27 Aktér
Zdroj: vlastní tvorba

Use case (typová úloha)

Use case představuje jednu ucelenou komunikaci aktéra se systémem. V rámci zapojení aktivit a **use case** můžeme hovořit, že 1 až N aktivit se podílejí na 1 **use case** (typové úloze). Naopak to ovšem neplatí. Jeden **use case**, může být rozšířen o další dílčí ***use case***, jako doplněk či nadstavba toho současného. Typové úlohy se pojí s aktéry a ostatními **use case** pomocí vazeb určující význam i samotných **use case**.

Obr. 28 use case
Zdroj: vlastní tvorba

Vazby

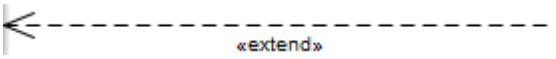
Spojnice mezi aktérem a **use case** či **use case** a **use case**. Tyto vazby doplňují význam samotných **use case** a aktéra. [7]

Use

Obr. 29 use Zdroj: vlastní tvorba

Vazba use se používá jako vazba mezi aktérem a **use case**. Tato vazba naznačuje, že daný aktér používá daný **use case**. Jeden aktér může využívat více **use case** a jeden **use case** může být používán více aktéry[7].

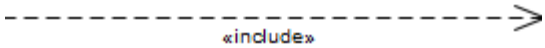
Extend



Obr. 30 Extend Zdroj: vlastní tvorba

Vazba **Extend** spojuje **use case** s dalším možným rozšířením typu **use case**. Vazba je směrem od možného rozšíření k rozšiřovanému. Rozšiřované nemusí nastat vždy, ale v rámci rozšíření se taktéž definuje bod v původním **use case**, kde může být spuštěno rozšíření. Dnes už méně používaná pro svou omezenou funkčnost, více používána místo **Extend** je vazba generalizace[7].

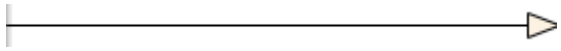
Include



Obr. 31 Include Zdroj: vlastní tvorba

Vazba **Include** je ve svém významu velice podobná předcházející vazbě **Extend**. Tato vazba rozšiřuje původní **use case** o další **use case**, zde ovšem rozšíření vzniká při každém spuštění hlavního **use case**. Vazba je vedena od původního **use case** k rozšiřujícímu **use case** a bod volání rozšíření se definuje ve scénáři hlavního use case. Toto rozšíření je například: Ověření bezpečnostního klíče, registrování zákazníka[7].

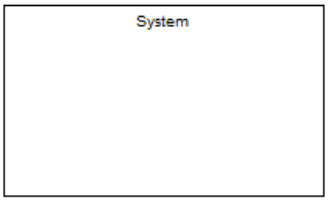
Generalizace



Obr. 32 Generalizace Zdroj: vlastní tvorba

Vazba generalizace slouží pro spojení dvou elementů stejného typu, jako znázornění vazby předka a potomka. Vazba mezi aktéry znázorňuje, že potomek používá všechny **use case** stejně jako jeho předek. K tomuto může používat ještě další **use case**, který předek nevyužívá. Při příkladu **use case** přebírá potom vlastnosti předka a přidává k těmto vlastnostem ještě další svoje vlastnosti [7].

Boundary (hranice systému)



Obr. 33 Boundary Zdroj: vlastní tvorba

Boundary slouží k rozdělení **use case**, které jsou implementovány v jednotlivých systémech. Jeden systém je znázorněn jedním **Boundary**. Může vzniknout situace, kdy jeden systém je aktérem pro druhý systém.

3.5 Enterprise Architect

Cabe a **case** nástroj **Enterprise Architect** slouží pro grafické mapování systému. Mohou být implementovány či vytvářeny v různých jazycích či platformách. Tento nástroj pomáhá při definici **Business** analýzy, při systémové analýze a datové struktuře systému. Taktéž poskytuje prostředí pro vytváření myšlenkových map a tvorbě webových či aplikací na android. Ve svém repertoáru nabízí širokou škálu modelů, ze kterých si uživatel může vybrat. Nástroj udržuje každou syntaxi i s jejich pravidly, i když nedohlíží na srozumitelnost či **Cirkulár** [12]efekt. Tento nástroj taktéž dopomáhá při týmových aktivitách v rámci diskusního prostředí **Gantt**[13] diagramu, rozdělení rolí, správných pravomocí či systému pro verzování. Všechny tyto přidané prostředky pomáhají k udržitelnému stavu projektu.

3.5.1 Verzování

Verzování je princip pro uchování určitého stavu systému v daném čase. Každá verze tak může nést vlastní označení, informace a cíle. Pro verzování jsou vytvořeny verzovací systémy, které soubory uchovávají a vytváří v nich určitou hierarchii podle čísla revize. Při každé revizi je uložen autor změn a čas změny. Takto můžeme vidět i celkový vývoj projektu od začátku tvorby až po jeho ukončení. Verzování se taktéž může hodit, pokud byl produkt v určitý čas nevyhovující a je nutné se navrátit k předchozí verzi. Tento princip je velmi důležitý a nutný při vývoji systému, kde se na samotném vývoji podílí několik desítek lidí a tak jejich aktivity v projektu musí být kontrolovány. Nejznámějším verzovacím systémem je například **Subversion**[14] **TortoiseSVN**, který je spolu se **SlikSVN** použit při návrhu v této práci[1].

---

[12] Cirkulár efekt – stav, kdy je systém zacyklen do smyčky
[13] Gantt diagram – znázorňuje řízení projektu v rámci času
[14] Subversion – je druh nástroje pro verzování souborů, pro každou zálohu se vytváří dokumentace loginu

4 Praktická část

Při modelování systému je optimální použít jednu z grafických syntaxí, které už v základu obsahují všechny potřebné elementy, aby obsáhly veškerý systém. Ovšem je tu další úroveň modelování, která není k dispozici lidem, kteří si nezaplatili náklady škole a semináře. Mnohdy na většině seminářů uslyší pouze základní informace a vytrženou část celku, jak se má postupovat, v modelování. Touto částí je například kooperace s lidmi na jednom projektu nebo základní rozdělení modelů **Packages**. Jaké máme mít optimální rozdělení systému, tak aby zůstal jednoduše čitelný, a přitom aby dokázal obsáhnout složitější systémy, jako jsou právě systémy **ERP**? Tento faktor je velice opomíjen a je dobré jej uvést na pravou míru, aby mohl být v budoucnu rozšířen a snad i používán. Spolu s modelováním konkrétního úseku systému je tento cíl nejdůležitější v této práci, a proto je na něj kladen ten největší důraz. V první řadě, tedy vymodelujeme systém do určité úrovně za pomoci syntaxe **BPMN**. V další části už takto vzniklý strom **Packages** porovnáme v prostředí **Enterprise Architect**. V této fázi nesmíme zapomenout na fakt, že zde nemůžeme myslet jako modeláři, ale i jako koncoví uživatelé nebo zákazníci. Tyto modely tak musíme přizpůsobovat i laickým čtenářům a proto v této úrovni nesmí být popis příliš technický.

Takto budeme postupovat i dále, kdy z **BPMN** vytvoříme **Use case** model a identifikujeme si možné kandidáty na roli třídy. Při konečné identifikaci poté můžeme zjistit, že každý úsek je jinak řešen a je důležité, abychom jej uchopili správně a zaměřovali svůj výklad už přímo na koncového uživatele. V případě koncového uživatele to může být zákazník či partnerská firma a v případě systémových **use case** a **class** to mohou být analytici vývojového týmu. Už pouze z tohoto úseku je možné vidět, že seřazení modelu může být víceméně velice složité.

Musíme dbát na fakt, že model je určen hned pro několik typů rolí, které přistupují k modelu systému jinak, a taktéž je přímo v modelu zajímají trochu jiné informace. Dalším faktorem, který bereme v potaz, je přístup hned několika pracovníků v rámci jednoho modelu. Při práci dvou či více specialistů na modelu mohou nastat nežádoucí chyby v podobě kolize úprav těchto pracovníků.

Třetím faktorem je úroveň modelování daného systému. V žádném případě nemůžeme modelovat systém do nejmenšího detailu, který daný systém může mít. Důsledkem takového přístupu by mohla být obtížná orientace v daném modelu. Míra informací by byla tak veliká, že by nikdo nemohl filtrovat přínosné informace a informace nepotřebné při práci v systému. Zde aplikujeme princip jednoznačného bodu, tzn., že budeme modelovat systém na takovou úroveň, kdy budou identifikované páteřní procesy a k nim přidružené procesy, které je jasně ovlivňují. Všechny procesy budou jasně identifikované a bude popsáno, pro jaké činnosti se daný proces používá. Kvůli jednoduchosti v procesech je také vhodné, abychom zjednodušovali procesy. Vytvářet komplexní diagramy na velikosti stran A1 je v dnešní době zastaralé. Taktéž se v modelu budou rozdělovat jednotlivé oblasti systému, ve kterých se daný proces bude vytvářet. Jelikož budeme modelovat Skladové hospodářství, bude prezentován princip těchto spojení a jeden základní diagram.

4.1 Mapování systému

Každý zkušený vývojář potvrdí, že pro správný vývoj systému je zapotřebí daný systém dobře znát. Zvláště v dnešní době, kdy se efektivita práce zvyšuje, se klade na vývojáře systému čím dál větší nároky v podobě požadavků na systém. Přístupy už 10 let staré jsou nahrazeny novějšími a propracovanějšími řešeními. Je na to dnešní svět připraven? Máme potřebné prostředky a snahu reagovat na tyto změny? Mnoho lidí si myslí, že když svou práci dělají stejně 5 let, tak budou tu samou práci dělat beze změn dalších 5 let. „Na co bychom něco měnili, když náš systém funguje a máme zákazníky?“. I takové odpovědi je možné slyšet v různých firmách. Většina takových lidí žije ve svých naivních představách o vlastní dokonalosti a nesnese pravdu o tom, že se svět kolem mění a staré přístupy je nutné nahradit novými. V dnešní době i systémy, které jsou dokonalé, stávají se za tři roky staršími variantami a musí se posunout dopředu, jinak se propadnou ve stále vzrůstající konkurenci[8].

Ovšem takovýto vývoj vede k dlouhé a strastiplné cestě dokumentování všech kroků v rámci systému. Mnohdy jsou textové dokumenty plné zbytečností, které nejsou pro vývoj a udržení moderního systému vůbec zapotřebí. Čím větší

systém je, tím obsáhlejší jsou dokumentace a člověk není schopen se v nich orientovat. Právě na tomto místě je nutné konstatovat, že i doba se připravila na takový spád a to pomocí grafických notací, které mohou dokonale zmapovat systém s využitím grafických syntaxí, jako například **BPMN, UML, Archimate** a další. Tyto syntaxe otevírají vývojářům, ale i uživatelům, obrovský svět, který není v dnešní době omezen žádnými překážkami. Tento grafický svět se nám otvírá, jak nám dokazuje i firma **Siemens PLM**. Výsledky této firmy jsou velice pokrokové a cíleně se zaměřují na trend, který bude za pár let. Doba papíru a neefektivní práce za čas vymizí, ve výrobě nahradí pozice lidí stroje a dokonce mohou pomáhat i v ostatních odvětvích. I když lidský faktor v pracovní činnosti nelze nahradit žádným dosud známým systémem, dokážou právě stroje zefektivnit práci nás lidí.

Potenciál je v simulacích a vytváření virtuálních objektů, které mohou testovat náš současný stav a tak můžeme sledovat i vývoj v budoucnu a upravovat podle těchto modelů výrobu a expedici. Simulace procesů bude ve firmách jednoho dne samozřejmostí a ten, kdo tento trend prospí, zaplatí velké náklady na vymodelování svých procesů. Bohužel se ještě dnes setkáváme s případy, kdy firmy samotné se při výhradně textové formě procesů zadrhávají na situaci, kdy ani ony samy neznají své vlastní procesy. Jak může firma fungovat, když pracovníci neznají firemní procesy? Každý zná svůj vlastní okruh práce a to mu stačí. Ovšem takové myšlení je v dnešní době zpátečnické, a je nutné jej nahradit kolektivní znalostí. Toho lze právě dosáhnout díky grafické notaci pro tvorbu procesních toků, jako je například **BPMN**.

Tato syntaxe je popsána v předchozí kapitole a z popisu je patrné, že poskytuje velký repertoár elementů pro modelování různých situací v systému. Díky této syntaxi se dají vymodelovat procesy do podoby jasně srozumitelné a redundantní. Díky přiloženému slovníku také můžeme překlenout most mezi slangem různých firem. V grafickém zamapování procesů se orientujeme daleko rychleji než v suchém textu plném nepotřebných informací. Tento způsob mapování ovšem také zlepšuje mnoho dalších procesů v samotném podniku.

Vezměme si **Change management** nebo **Incident management**. Tyto dva postupy se musí vyznat ve stávajících procesech, když hledají správný způsob pro změnu současných procesů nebo hledají důvod, proč daný proces přestal fungovat.

V dobře zmapovaných systémech to zabere příslušnému pracovníkovi velmi krátkou chvíli. V textové formě nebo ve vůbec žádné formě se tyto problémy mnohdy nikdy nevyřeší a jen se to po nějaké době zametou do koše. Je tento postup správný? Není lepší vyřešit všechny vzniklé incidenty a problémy, aby se neopakovaly? Následující kapitoly nám mohou nastínit tvorbu našich procesů v rámci firmy tak, abychom je doopravdy znali. I když si říkáme, že je známe, je tomu i u našich spolupracovníků?

Nechápeme každý z nás daný proces trochu jinak? I z maličkosti na začátku se může vyklubat obrovský problém na konci vývoje systému. Efektivní práce je sjednotit terminologii uvnitř firmy, abychom věděli všichni stejně, jaké procesy máme, jak se chovají a jak je můžeme upravit tak, abychom dosáhli požadovaného cíle. Už méně si lidé uvědomují další přínosy takto vymodelovaných procesů. Například při školení nových lidí, jaké máme příručky? Mnohdy sedí nový zaměstnanec i několik měsíců v kanceláři, sbírá si všechny informace sám, mnohdy ani nemá odkud a zaměstnavatel čeká, že se na daném místě zaškolí velmi rychle. Ale jak? Když přechází člověk z odvětví do odvětví, nemá zkušenosti, musí začít, ale nemá z čeho. A právě v tomto může pomoci i grafické mapování. Díky namodelovaným procesům můžeme identifikovat všechny aktivity, kterými se nový pracovník bude zabývat, a tak i určit celý okruh zodpovědnosti tohoto pracovníka. Navíc když se tento pracovník bude zabývat prací v systému, díky **use case** modelu může být zmapována oblast a stanoveno jaké funkce v daném systému bude obsluhovat. Scénáře napoví, jak daný systém obsluhovat a díky pár zkouškám si osvojí práci velmi rychle a snadno. Díky těmto krokům se výrazně sníží náklady na zaškolení pracovníka a jeho začlenění do běžného chodu je velmi rychlé a hlavně efektivní. Dalším pozitivním přínosem tohoto grafického mapování je školení v rámci našich zákazníků[8].

Pokud jsme vývojářská firma a zajišťujeme školení pro uživatele našeho systému, jak bychom mohli lépe seznámit uživatele s celým procesem, než mu přesně ukázat jaké jsou odlišnosti od standardního procesu nebo jaké nové aktivity a funkce byly přidány v rámci požadavků do systému? Právě pomocí těchto procesů a use case můžeme vše jasně ukázat a úspěšně vyškolit dané pracovníky. Navíc můžeme tyto závěry spolu s grafickými typovými úlohami a procesy

poskytnout účastníkům školení jako ucelený manuál. Díky tomuto materiálu si mohou uživatelé vybavit postupy, které se na školeních probíraly. V neposlední řadě jsou takové metody moderní a v současné době nutné[8].

4.2 Nástroj a struktura projektu

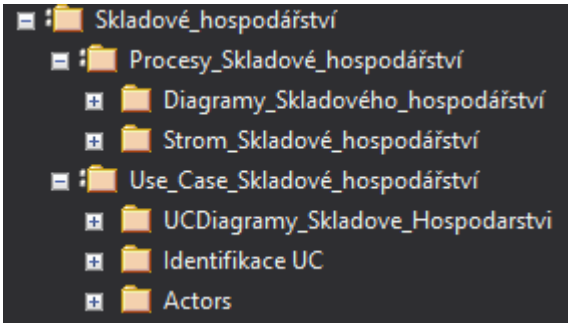
Pro vypracování projektu byl zvolen **Case** nástroj **Enterprise Architect**. Tento nástroj je zmíněn v předchozí kapitole. Tento nástroj podporuje dokonalé prostředí pro kooperaci více uživatelů a nabízí uživateli přívětivou tvář při vytváření modelů. Přesněji se používá **Enterprise Architect** verze 12. Každý projekt potřebuje pro svůj vývoj dobré prostředí a kvalitní rozvržení jednotlivých kroků. Proto, aby byl projekt dobře čitelný, a do jisté míry jednoduchý, musí se přizpůsobit i hlavní strom projektu. Tento strom se nazývá **Project Browser** a slouží pro členění modelů do struktury modelu za pomoci **Package** (balíčků či složek). Když máme jednoduchý projekt, čítající jen pár diagramů, tak nás tento strom moc nezajímá. Ovšem když počítáme s větším projektem, čítajícím i několik desítek modelů a oblastí, je zapotřebí si v něm udělat systém, abychom se v něm neztráceli. Pro naše účely, kdy se jedná o jednu oblast **ERP** systému, jsou nároky na tento strom velké. Požadavky na uzpůsobení stromu musí být jasně specifikované a dané, jinak při změně požadavků v průběhu modelování, když už náš strom naroste do velikosti, by jeho přepracování znamenalo velkou časovou ztrátu. Kvůli těmto skutečnostem je nezbytné, aby byly požadavky jasné. I když při vývoji vyplývají na povrch další pohledy, které bychom chtěli mít na daný systém, budeme aplikovat právě zmíněné požadavky, které je zapotřebí nachystat v **Project Browseru**, abychom tam mohli doplňovat pouze diagramy a elementy. Základní požadavky na modelování tohoto projektu jsou takové.

- Základní rozvržení procesů do jednotlivých **Business** procesů
- Vymodelování procesů až na atomickou úroveň
- Doplnění datových typů do procesního modelu
- Vymodelování základních typových úloh a identifikace jejich místa v procesu.
- Rozvržení základních stavů

- Přístup pro více uživatelů zároveň
- Schopnost projektu zabránit kolizi víceuživatelského přístupu
- Generování dokumentace z daného projektu

Na základě těchto požadavků si stanovíme základní úrovně **Package** dle nejvyšších úrovní **Business** procesů, které nadefinujeme v dalších podkapitolách. Výsledný strom v **Project Browseru** si tedy připravíme. Vždy se musíme zamyslet nad otázkou, zdali do daného projektu nebudeme přidávat další oblasti, modely a digramy. V rámci všech těchto podmínek lze připravit **Project Browser** takto:

Z obrázku je patrné, že pod nejvyšším Modelem je složka Skladové hospodářství, která je vrchní složkou pro procesní mapování a **Use case**. Procesy jsou dále rozděleny na diagramy a Strom.



Obr. 34 Strom Projektu Zdroj: vlastní tvorba

V **Package** procesy se budou ukrývat diagramy jednotlivých procesů a úrovní. V **Package** Strom se budou ukrývat vyšší pohledy na úrovně diagramů. V **Package Use case** se prvně identifikují **use case** a poté se přetvoří do standartních diagramů jazyka **UML**. Identifikace bude za pomocí aktivit. Z tohoto stromu taktéž budou vycházet ještě další dílčí balíčky, které budou rozšiřovat tyto **Package**.

4.3 Víceuživatelský přístup

Při práci na rozsáhlých projektech mnohdy kooperuje více uživatelů najednou. Je tedy nezbytné, aby nám nástroj dokázal pomoci, abychom mohli spravovat systém bez rizika kolize, které se zvyšuje při větším počtu účastníků. Jak ovšem docílit tohoto bezpečnostního opatření? Je několik možností, jak toto opatření uskutečnit.

4.3.1 Možné varianty

První je rozdělení modelů a pomocí **Control Package** si vytvořit jedno pracovní místo (na serveru, **Cloudu** či podobném uložišti) a ukládat sem **XML**.

Každý by měl pouze jeden **Control Package** u sebe a spravoval by si svou vlastní oblast. Při vytváření projektu, kdy je zapotřebí mít všechny oblasti spolu, jednoduše naimportujeme všechny požadované oblasti dohromady. Na druhou stranu, je zde problém v tom, že pokud mají oblasti společné **Objecty**, není možné je spravovat z vícero míst v rámci tohoto řešení.

Druhým způsobem, jak spravovat týmovou verzi projekt, je sdílení jednoho projektu v rámci funkce replikace, kterou nabízí **Enterprise Architect** v základu. Při tomto způsobu je nezbytný vedoucí projektu, který dává svolení pro synchronizace replik s jeho originálem. Při jakékoliv změně se může provést synchronizace hlavního projektu s jeho replikami a tak se dostávají informace všem, kteří mají replikovanou verzi projektu. Tento přístup ovšem předpokládá velkou kontrolu vedoucího projektu, jelikož při synchronizaci se může stát, že při změně jednoho elementu na dvou místech dojde k přepsání pouze jednou variantou a druhá varianta je smazána. Proto je zde kladen velký důraz na řízení projektu.

Třetím řešením je tvrdá politika, která by jakýkoliv přečin trestala a vyžadovala opravení stavu. Tento stav je už přežitý a naprosto neadekvátní.

Čtvrtým způsobem je použití **SVN**, neboli **Subverison**. Tento verzovací systém může být plně kooperující s **Enterprise Architektem** a může nejen vytvářet verze, ale taktéž vytvářet dobrý protikolizní systém. Na druhou stranu toto řešení přináší změny, které je třeba před používáním realizovat, a to vytvoření či zakoupení **SVN** serveru a verzovacího nástroje. Právě tímto třetím principem se budeme zabývat a bude zde popsán přesný postup, jak se tímto doplňkem nechat vést, abychom daný projekt dotáhli do vítězného konce.

4.3.2 Subverison

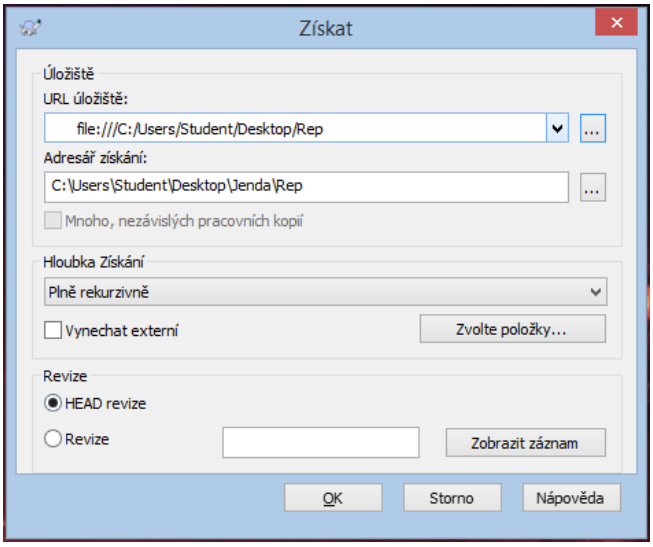
Tento nástroj je vysvětlen v teoretické části této práce. Jakým způsobem se vnáší do prostředí **Enterprise Architect**, a jak je poté používán spolu s tímto nástrojem, bude vysvětleno nyní. Tyto informace je obtížné nalézt na internetu v českém jazyce. Proto může být tento postup používán jako vzor pro práci ostatních firem v jejich prostředí **EA**. Mnohdy jsou tyto informace soustředěny do konkrétních školení ohledně **EA** a zpoplatněny značnými sumami.

Příprava

Před používáním je nutné mít **Subversion** na svém počítači. Jaký ovšem použít? Když nepotřebujeme složitý a finančně náročný nástroj, bude nám stačit **Subversion** s názvem **TortoiseSVN**. K tomuto nástroji byla vydána sbírka základních funkcí v českém jazyce, takže je přístupný všem a ukazuje názorně, jakým materiálem se může člověk nechat vést, aby tento nástroj používal přesně. V případě zájmu o tento nástroj je možné jej nalézt ve zdroji [15]. Tento nástroj je volně ke stažení a podporuje free verzi pro práci. Spolu s tímto nástrojem je možné pro práci s **Enterprise Architect** použít i doplněk s názvem **SlikSVN**. Taktéž tento nástroj je potřebný nainstalovat. Když máme oba dva programy nainstalované, máme vše připravené.

Konfigurace s nástrojem

Před napojením **Subversion** na **EA** potřebujeme vytvořit repository. Do těchto repositorů se nám budou ukládat všechny verzované **Package**. Je dobré si ještě nainstalovat češtinu pro daný verzovací systém. Po instalaci **Tortoise** do počítače si nyní vytvoříme složku v adresáři, kde chceme ukládat náš projekt. Pravým klikem na danou složku >>> **TortoiseSVN** >>> Vytvořit úložiště zde. Nyní si vytvoříme druhou složku, kde budeme mít současný projekt. Právě tato složka se napáruje na dané repository, a co bude v této složce, bude i v repositorech. Vytvoří se obdobným způsobem, jako repository. Pravým kliknutím na vybranou složku >>> **SVN** Získat …, se nám otevře toto okno: obrázek č. 35. Do **URL** úložiště zadáme cestu k repositorům a Adresář získávání se nám nastaví podle současného místa vybrané složky. Hloubku získávání zvolíme „plně rekurzivně“ a potvrdíme.

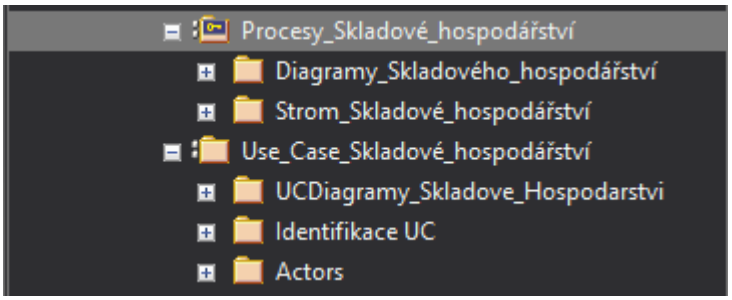


Obr. 35 Pracovní složka Zdroj: vlastní tvorba

Nyní si otevřeme náš budoucí projekt v **EA**. Buď dříve vytvořený, nebo si vytvoříme nový. Zde si nastavíme verzi. V menu >>> **Project** >>> **Version Control** >> **Version Control Settings**. Zde vybereme v **Check boxu Subversion** a zadáme unikátní číslo. V kolonce **Copy Path** zadáme cestu ke složce, která je napárována na repository a do **Subversion Exe path** zadáme cestu v **Program files**, kde máme uložený **SlikSVN**. Podobná cesta jako tato: ***C:\Program Files\SlikSvn\bin\svn.exe***.

Vše uložíme tlačítkem **Save**. Taktéž můžeme vytvořit několik různých míst s repository, které budeme mít napojené na jeden soubor **EAP**. To vše záleží na nás.

Nyní si v **Project Browser** vybereme **Package**, který chceme naverzovat. V praxi se vytváří na každé úrovni, kde může dojít ke kolizi.



Obr. 36 Zámek Subversion Zdroj: vlastní tvorba

Tento **Package** vybereme pravým klikem na daný **Package** >>> **Control package** >>> **Configurate**. Zde zaškrtneme „**Control Package**“ a ve **Version Control** vybereme podle unikátního čísla naši verzovací složku. Pokud chceme jinak nazývat **XML**, můžeme název změnit dle vlastního uvážení. Většinou ovšem **XML** přebírá název **Package**, pro který byl tento soubor vytvořen. A vše potvrdíme. Nyní se nám ukazatel **Package** změnil. Na obrázku složky se nám vytvořil klíč, pokud jsme nezaškrtli **Keep Out**. Pokud jsme tento **Check box** zaškrtli pravým tlačítkem na daný **Package** >>> **Control Package** >>> **Check in**. Tak se nám uloží daný **Package** do formy **XML** do námi připravené složky. To si můžeme zkontrolovat otevřením složky, která je na pojená na repository.

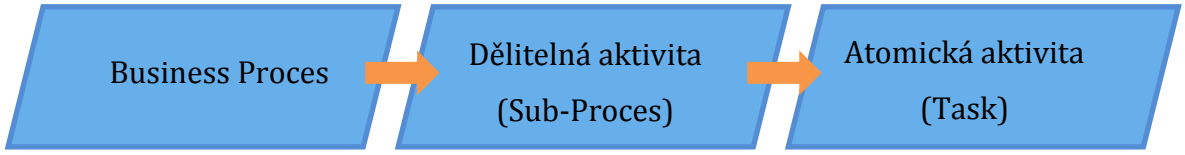
Jak nám to pomohlo s víceuživatelským přístupem?

Na první pohled nám to při víceuživatelském přístupu nikterak nepomohlo. Máme naverzovaný projekt, ale co vlastně přesně znamená ten zámek na package? Tento zámek znamená, že je **XML** daného package připraven pro otevření jednoho uživatele. Když se zámek otevře, zámek zmizí a to znamená, že pouze my máme přístup do daného adresáře. Nikdo jiný v něm nemůže pracovat a provádět změny.

Po dobu naší práce tak máme chráněná data. Až skončíme se svou prací, jednoduše uzamkneme **Package**. Poté si jej může otevřít další uživatel a pracovat na něm on. Avšak toto nastavení je funkční až ve chvíli, kdy se linkovaná data nacházejí na serveru, kde mají ostatní stakeholdeři přístup. Pokud pracujeme ve skupině, většinou pracujeme na sdílených serverech. Jeden z těchto serverů může být typu **SVN**, kde vytvoříme repository. Následně na jiný nebo stejný server či složku už na lokálním počítači si vytvoříme složku napojenou na tyto repository. Když vytváříme odkaz na plné repository, tak se nám již existující složky v repositorech promítnou do nalinkované složky. Vše ostatní se nastaví stejně. Můžeme mít taktéž zaheslovaný přístup na servery, a proto se na naše data dostane pouze ten, kdo dostal přístup. Všichni ostatní nemohou projekt měnit. Tento postup je velmi účinný. Pro přesnější představu, jak se s tímto **SVN** nástrojem pracuje, je dobré se seznámit s dokumentem pojednávacím o verzovacím nástroji **ToritoiseSVN,** který je přiložen ve zdrojích pod číslem [15]. Tyto přístupy je vhodné aplikovat na začátku projektu nebo se dají doplnit zpětně do již vytvořeného projektu. Výhody tohoto postupu jsou zřejmé. Máme kontrolu nad celým projektem i nad jeho úpravami. Navíc při nečekané chybě můžeme identifikovat uživatele, který tento problém způsobil, jelikož se ukládají i **Loginy**, které vstupují do daného **Package**. Tento postup aplikujeme i v příkladu Skladového hospodářství níže.

4.4 Jak tvořit grafické mapování

V této kapitole se proberou postupy, kterými je dobré se řídit, aby se námi vymodelovaný systém tvořil snadno a jasně.



Obr. 37 Princip modelování Zdroj: vlastní tvorba

Páteřní procesy

Při modelování **BPMN** se při zpětném modelování systému začíná vždy od páteřních procesů. Tyto procesy jsou na nejvyšší úrovni a obsahují v sobě **sub-procesy**, které rozvíjejí význam **Business** procesu. Tyto **Business** procesy mohou

být i procesy, které ohraničují jednotlivé úseky systému, které se prodávají samostatně. V příkladu systému, který je zde vytvořen, jsou **Business** procesy, jako druhé nejvyšší procesy a jsou obsaženy v nejvyšším **Business** procesu Skladové hospodářství, které je určeno jako oblast pro prodej.

4.4.1 Sub-procesy

Nyní se identifikují procesy od nejvyšší úrovně směrem dolů. Takto můžeme vymodelovat několik úrovní hierarchických **sub-procesů**, nežli budeme modelovat samotný tok. Při takovémto jemném rozlišování jednotlivých úrovní může nastat problém. Pro jednoznačnost by měly být (pokud to situace nevyžaduje jinak) modely jednodušší ke čtení, protože přílišné větvení může být spíše ke škodě.

4.4.2 Atomické aktivity

Při vytváření nižších úrovní procesu jsou procesy, které mají svůj tok a zároveň obsahují ještě další vnořené **sub-procesy**. V takovém případě je nutné definovat tok procesu a další vnořené **sub-procesy**. Další vnořené **sub-procesy** modelujeme do nejnižší úrovně, která je nutná pro pochopení systému. Například proces níže „zabalení produktu do obalu“ je proces, ve kterém se veškerá činnost dělá bez přítomnosti systému. V určitém případě je zbytečné modelovat právě tyto modely. Nikterak nám nepomohou pro pochopení systému. Ovšem pro pochopení vnitřních procesů podniku, je to nezbytné. Například pro optimalizaci systému nebo pro školení dalších pracovníků, či řešení problémů v **Incidentu** managementu je to nedostižný pomocník.

Jak vidíme na obrázku č. 41, v první řadě se vytvoří **Business** procesy, které se upřesní **sub-procesy**. Ty jsou následně popsány až na úroveň atomických **Tasků**.

4.4.3 Jak vytváříme tok procesu?

Kvůli otevřenosti syntaxe **BPMN** je obtížné definovat jednotnou metodiku pro všechny uživatele dané syntaxe. Tato metodika je ovšem nutná, takže bude nadefinována v menším měřítku a se základními pravidly. Díky této metodice

ovšem mohou být zpracovány procesy jednotně, a tak při komunikaci dvou uživatelů stejné metodiky dojde k vzájemnému pochopení.

Název aktivity

Aktivita musí být pojmenována určitou činností. Například „Vyskladnění“. Jméno má reflektovat, co se v dané aktivitě dělá, proto jeho název by měl být co nejvíce vypovídající.

Popsání aktivit

Každá aktivita musí být popsána bud ve formě **Linked Documentu** nebo popiskem. I když se jedná o dobře známou aktivitu, je zapotřebí ji popsat.

Postavení diagramu

Diagram se modeluje od levého horního rohu směrem dolů. Diagram je tedy vertikální. Je to z důvodu nejoptimálnějšího umístění na stranu A4. Pokud máme tok, který je méně členěný ve smyslu dělení toků, je vhodné modelovat diagram vždy z levého rohu do pravého rohu jedné stran A4, která je znázorněna pomocí šedé čáry v prostoru pracovní plochy.

Začátek diagramu

Diagram **Workflow** je zahájen událostí, která spustí daný proces. Událostí, které spustí jeden proces, může být několik, vždy ale musí být alespoň jedna. Při ukončení daného procesu je taktéž zapotřebí modelovat i ukončovací událost. Může nastat situace, kdy jedna ukončovací událost odesílá data pro začátek jiného procesu.

Vnořování aktivit

Pokud máme **sub-proces**, který obsahuje jen pár elementů, je vhodné jej vytvořit jako **sub-proces „Is Expanded“**. Tzn., že jeho tok bude na vyšší úrovni viditelný. Limit takto vytvořený je 5 elementů dohromady. (Tzn. Elementy jsou události, aktivity, pool, …)

Vytvoření slovníku

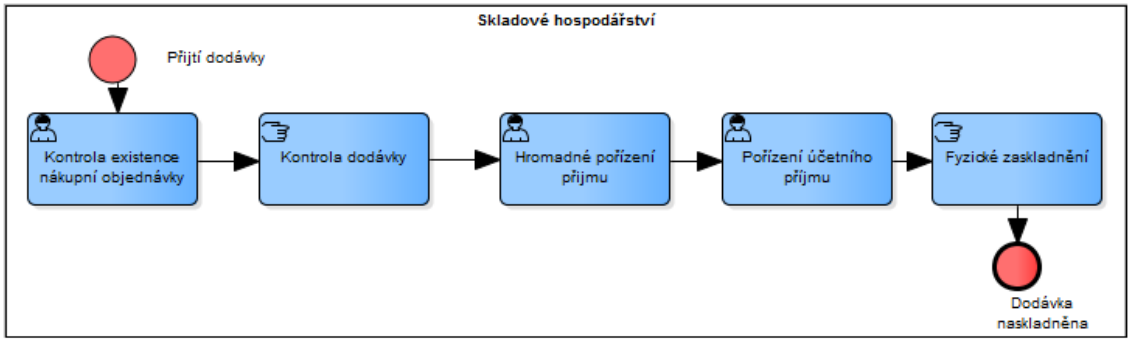
Pro jednoznačnost potřebnou pro jednání s jinými firmami je nutné vytvořit slovník, kde definujeme naše a jejich slovní obraty. Díky těmto krokům se rozdíly mezi námi a ostatními eliminují a vytváří se prostředí pro komunikaci.

Další připomínky

- Do jednoho diagramu se modeluje jeden model
- Neponechávat objekty bez vazeb
- Neměnit velikost aktivity, pokud to není nutné
- Snažit se o nekřížení stejných vazeb, pokud tento bod nelze dodržet, za pomoci barev odlišit křížící se vazby
- Pokud uvádíme zkratku v názvu elementu, její plný název musí být obsažen v popisku příslušného elementu

4.4.4 Happy flow

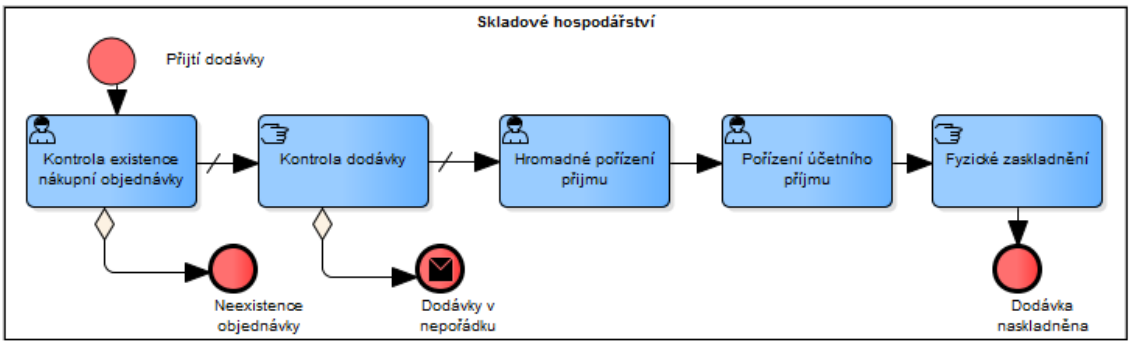
Při vytváření toku mějme na paměti, že proces se vždy vytváří od tzv. **Happy flow**. Tento termín značí proces, ve kterém nejsou negativní podmínky. Vždy tedy dosáhneme cíle. Tento postup je stejný, jako při vytváření scénářů v use case, kdy vytvářím **Happy** scénář **Basic**. V tomto scénáři popisujeme také vždy pouze ty kroky, které nás zavedou k cíli, bez výjimek. Výjimky se ve scénáři píší do **Alternate** nebo do **Exception** scénáře. To samé je i u vytvářených procesů. V prvé řadě si vytvoříme hlavní tok a poté vytvoříme všechny alternativní cesty, které mohou nastat v procesu. V praxi je vhodná demonstrace na příkladu. Mějme proces Příjem z nákupu hromadný. Při přijetí se provede kontrola existence objednávky, zkontroluje dodávka, vše se hromadně přijme a účetně navede do systému. Poté už stačí jen zaskladnit a máme celý proces hotov, viz obrázek č. 38. (Tento proces je více popsán níže v celkovém příkladu na Skladové hospodářství)



Obr. 38 Happy Flow Zdroj: vlastní tvorba

Nyní je nutné si uvědomit, že v případě, že jedna z první kontrol prokáže nesrovnalost, proces tím skončí a proces neskončí zaskladněním dodávky, tzn., že

se musí vyřešit nastalý problém. Touto výjimkou může být například odeslání objednávky zpět dodavateli nebo přijetí pouze produktu, který souhlasí s naší objednávkou a ostatní se vrátí zpět. Nebo se proces ukončí a vzniklý problém vyřeší obchodní oddělení. V našem případě berme v potaz, že vzniklý problém vyřeší obchodní oddělení, takže pouze se ukončí tok. Viz obrázek č. 39.



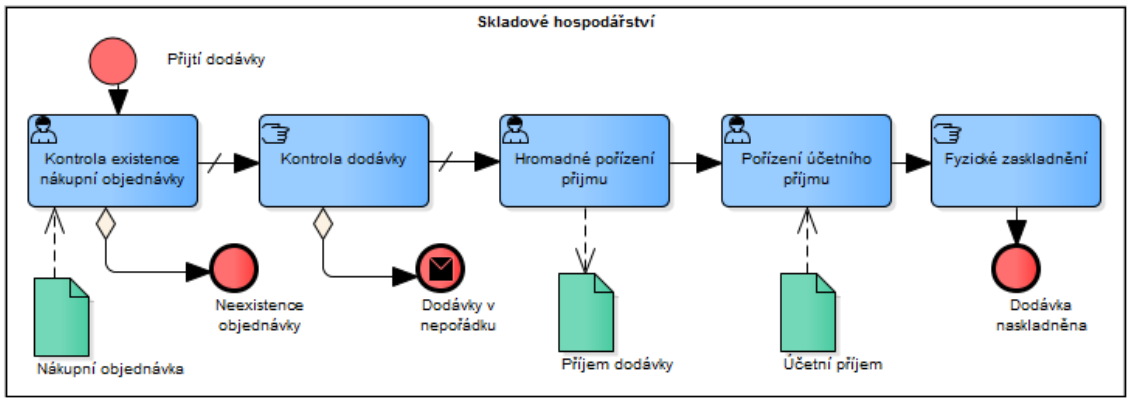
Obr. 39 Přidání alternativ Zdroj: vlastní tvorba

V toku nastaly změny. Ke kontrole existence nákupní objednávky se přiřadil end event. Ke kontrole Dodávky manuální se přiřadil **End Event** typu **Message**, který odešle zprávu o zjištěné chybě při kontrole. Je nutné říci, že se zde nepoužil klasický **Gateway Exclusive**, ale **Sequence flow** typu **Conditional**. Takto nemusíme přímo zasahovat do našeho modelu a zjednodušíme si úpravu procesu. Taktéž se nám změnily toky z aktivit kontrol na typ **Default**, takto se v případě podmínky tok nebude ubírat dál, ale půjde pouze tokem s podmínkou. Takto vytvořený proces už splňuje všechny předpoklady a je tedy připraven k představení a mohou se s ním seznámit všichni uživatelé a pracovníci vývoje.

4.4.5 Doplnění Diagramu o Data Objects

Nyní, když je vyhotoven proces v pohledu aktivit a alternativních cest, je dobré přiřadit i datové objekty, které v diagramu mohou figurovat. Tyto datové objekty odráží datové toky či objekty, které se vytvářejí či používají v rámci procesu z některých aktivit. K těmto datovým objektům se přiřadí pouze směrová vazba, která bude ukazovat, zdali je daný objekt vstupující či vystupující. I když má syntaxe možnost rozlišovat datové objekty na vstupující a vystupující, tak toto řešení není vhodné při velkém systému, kdy jeden datový objekt je vstupující v jednom diagramu a vystupující v druhém objektu. Zde může vzniknout námitka, proč se neudělají dva datové objekty. Tato myšlenka je sice na místě, ale musíme si

uvědomit, či při správě tak velkého modelu celého systému se mohou upravovat popisy jednotlivých datových objektů. Udržovat tyto popisy je velmi obtížné a proto je lepší, když se popisy upravují pouze na jednom místě, nežli v několika datových objektech stejného jména a charakteru. Příkladem může být předchozí diagram, který jsme si namodelovali. Výsledný model i s datovými objekty vypadá následovně.



Obr. 40 Doplnění diagramu Zdroj: vlastní tvorba

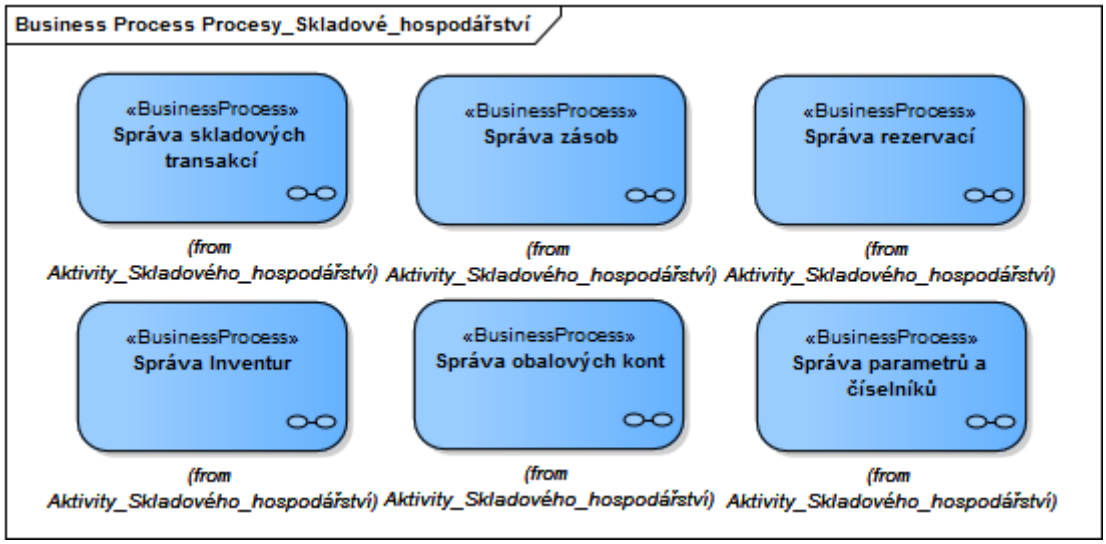
4.5 Skladové hospodářství – BPMN

Pro účely této práce byl použit jako vzorový příklad skladový systém, který je v širším tématu rozebrán ve třetí kapitole. Tento systém bude rozebrán až na nejnižší úroveň. Všechny kroky budou popsány a definovány. Podle tohoto příkladu poté mohou čtenáři získat základní představu, jak by se mělo postupovat při zpětném modelování systému. Také jej mohou použít jako vzor pro modelování procesů ve svých firmách. I když zde není možnost rozepsat celý průběh detailního postupu při modelování, může být tento rámec použit jako vzor. Je jasné, že pro zmodelování celého skladového hospodářství by nám daný rozsah této práce nestačil. Proto byly zmapované vždy jedny varianty daných procesů. Při všech variantách by bylo od každého procesu minimálně 10 variant. Každý systém či výrobce má své postupy.

4.5.1 Páteřní procesy

Ve skladovém hospodářství tedy musíme hned na nejvyšší úrovni rozlišovat několik zásadních kapitol, které mohu být ke standardnímu skladovému hospodářství přidruženy, a tak i rozšiřovat jejich funkčnost.

Podle obrázku č. 41. se identifikovalo celkem 6 **Business** procesů, které budou hlavními úseky oblasti Skladového hospodářství. Barva je zvolena záměrně kvůli jednoduší identifikaci procesů. Tento důvod bude popsán níže.



Obr. 41 Páteřní procesy Zdroj: vlastní tvorba

Nejdůležitější **Business** proces ve skladovém hospodářství je bezpochyby Správa skladových transakcí. Tento **Business** proces už v názvu naznačuje, že se zde budou modelovat procesy související se skladovými pohyby. Bude mít v sobě transakce potřebné pro vyhotovení objednávek, přesun produktů na výrobu, či uskladnění nebo vyskladnění.

Dalším **Business** procesem je Správa zásob. Tento **Business** proces bude sloužit pro správu požadavků a jejich vypracování právě pomocí business procesu Správa skladových transakcí.

Pro vypracování objednávek je také důležité, aby byly produkty řádně rezervovány pro zákazníka, případně pro prominentní zákazníky chceme, aby jim byly dodány produkty v co nejkratším čase.

V každém skladu jsou neméně důležité inventury, které se provádí každý rok a podle jejich výsledku se zjišťuje skutečný stav zásob na skladě.

Nemohou být opomíjeny ani obaly a jejich cesty s produkty. Mohou to být klasické obaly, které jsou použitelné jednou, a tudíž sledujeme pouze stav našich obalových zásob a jejich vydávání. Zde jde pouze o čísla. Poté jsou zde vratné obaly, které jsou potřeba pro balení speciálních produktů například pro polotovary

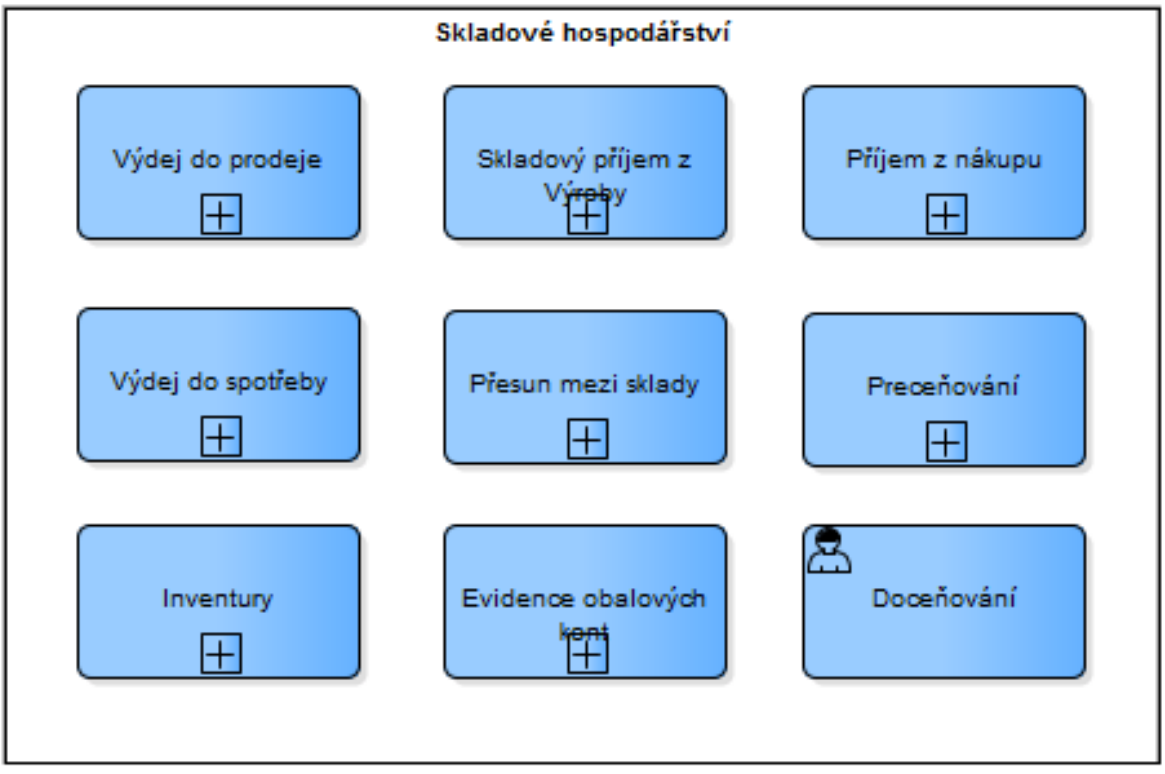
strojů. U těchto obalů musíme kontrolovat celý cyklus, jak odesílání s patřičným produktem, pro který byl obal určen, tak i jeho příjem zpět k nám na sklad.

Jako poslední **Business** proces, který je důležitý pro správný chod systému, je Správa parametrů a číselníků. Aby systém správně fungoval, musí být všechny číselníky a parametry naplněny.

Po vytvořeních páteřních procesů můžeme vidět, že na nejvyšší úrovni se skladové hospodářství zabývá právě těmito nejvyššími procesy. V tuto chvíli nás tak pouze zajímají oblasti, ve kterých se skladové hospodářství angažuje. Při definování této úrovně, i když známe všechny nižší úrovně a obecně jejich procesy, musíme vyfiltrovat pouze takové informace, které vedou k identifikaci této nejvyšší úrovně.

4.5.2 1. Úroveň

Po identifikaci nejvyšší úrovně je nyní na řadě definování další úrovně procesů daného systému. Nejvyšší úroveň byla business procesy a nyní již budeme přesně specifikovat **sub-procesy**, typu aktivity.



Obr. 42 První úroveň Správy skladových transakcí Zdroj: vlastní tvorba

Správa skladových transakcí

Tento **Business Process** nese všechny skladové manipulace, které se ve skladovém prostředí definují. Ke každému elementu je samozřejmě potřebný příslušný stručný popis, takže bude každá aktivita následně popsána.

Výdej do prodeje

Tato aktivita slouží pro transakční operace, kdy se dané množství produktu vyskladní pro prodejní účely. Zde se budou vyskladňovat produkty za účelem vyhotovení objednávky pro zákazníka.

Skladový příjem z Výroby

Tato aktivita slouží pro manipulace produktů z výroby. Vyrobené produkty jsou zaevidovány a uskladněny na příslušné umístění.

Příjem z nákupu

Skladové procesy, kdy nakoupené produkty jsou v systému zaznamenány a uloženy na příslušné místo ve skladu. Přijaté produkty musí být zkontrolovány s objednávkou, která byla vytvořena námi pro dodavatele.

Výdej do spotřeby

Proces určen pro popsání všech transakcí, při kterých se vyskladňují produkty ze skladu a jsou převezeny na příslušné výrobní místo, kde bude daný produkt zpracován dalším postupem.

Přesun mezi sklady

Proces určující přesný postup pro přesun jednoho produktu mezi sklady. Typy skladů a jejich účel jsou popsány v teoretické části a proto zde nebudou rozebírány.

Přeceňování

Proces určující postup při změně hodnot produktu. Při nových cenách produktu musí dojít k přecenění všech produktů, které jsou na skladě, nebo produktů, které jsou pro toto přeceňování určeny.

Inventury

Proces manipulace při procesu Inventury. Přesně určení množství produktů, které jsou na skladě.

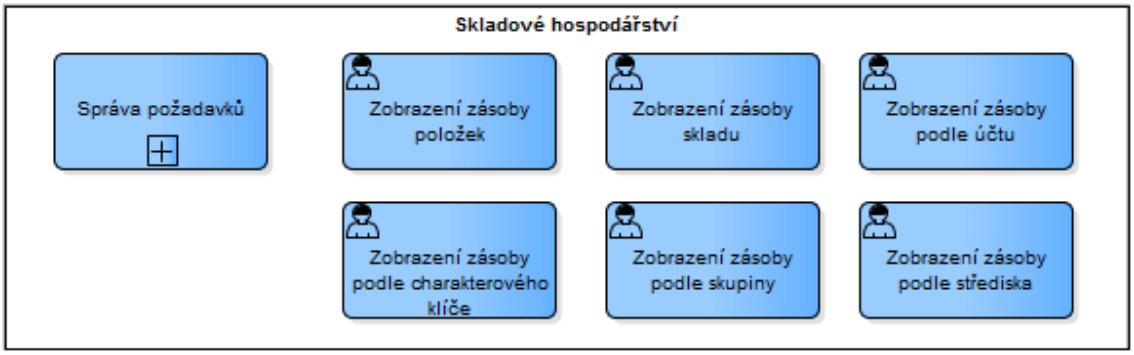
Evidence obalových kont

Příjem a výdej obalových prostředků pro vyhotovení objednávek. Postupy pro přijetí obalů z nákupu či od dealera obalů a výdej obalů do spotřeby, kdy se vyskladní pro zabalení příslušného produktu do obalu.

Doceňování

Proces, kdy se doceňují výrobky zpětně či jiným způsobem.

Správa zásob



Obr. 43 První úroveň Správy zásob Zdroj: vlastní tvorba

Jak bylo zmíněno výše, jedná se o Business proces, který se stará o zásobu a její vztah k požadavkům. Je tedy zodpovědná za vyhotovování požadavků a pohled na jednotlivé zásobovací produkty, které jsou na daném skladě.

Správa požadavků

V této aktivitě budeme řešit příchozí požadavek, který je vytvořen například pro vyskladnění produktu za účelem vyhotovení objednávky č. 43. nebo pro přesun produktu ze skladu do výrobního prostředí a podobné funkce tohoto požadavku.

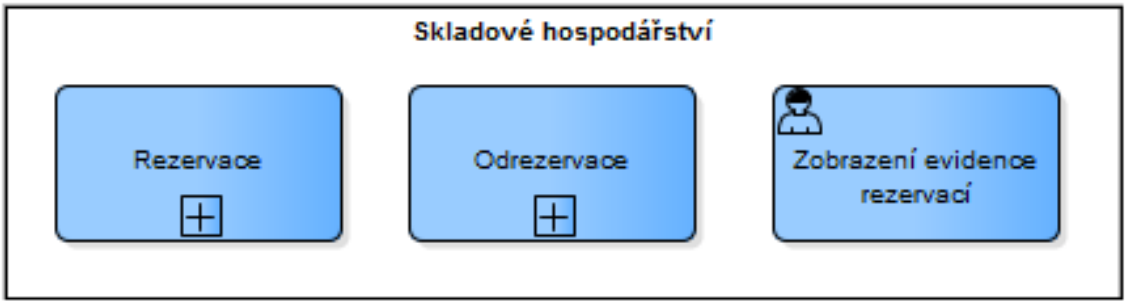
Zobrazení

Zde vidíme první atomické aktivity, které se vyskytly v našem základním systému. Jsou vytaženy na vyšší úroveň pro rychlejší a jednodušší pohled uživatele na model.

Tyto aktivity slouží pro vhled do zásob podle určitého filtru. Abychom specifikovali přesně podle jakých filtrů, je zde rozepsáno šest základních atributů. Můžeme si tak zobrazit všechny položky zásoby bez filtru nebo pouze položky zásob na určitém skladu, podle skladového účtu, charakterového klíče, skupiny či

podle střediska. Tyto aktivity jsou upřesněny jako **User Task** (popsáno v kapitole 3.2.1.).

Správa rezervací



Obr. 44 První úroveň Správy rezervací Zdroj: vlastní tvorba

Procesy, zabývající se rezervováním a odrezervováním množství produktu na určitou objednávku nebo zákazníka. Taktéž můžeme evidovat jednotlivé dílčí aktivity.

Rezervace

Procesy zabývající se zarezervováním určitého množství produktu na příslušnou objednávku nebo zákazníka. V určitém momentu je zkontrolován stav zásob na skladě, a pokud je splněna podmínka, aby bylo uskladněné množství určitého produktu větší jak množství požadovaného množství k rezervaci, je dané množství zarezervováno a tím pádem i vytaženo ze zásob produktu. Takto zarezervované produkty nejsou k dispozici k další manipulaci pro jinou objednávku nežli pro tu, pro kterou je dané množství produktu zarezervováno.

Od-rezervace

V případě zrušení objednávky nebo změně priority při vychystání produktu, může dojít k odrezervování daného množství produktu. Tím pádem jsou dané produkty znovu zařazeny pro další manipulaci nebo jsou opět přímo zarezervovány pro další objednávku.

Zobrazení evidence rezervací

Uživatel si může prohlédnout stav rezervací v příslušném formuláři, kde může provádět další úkony spojené s rezervacemi.

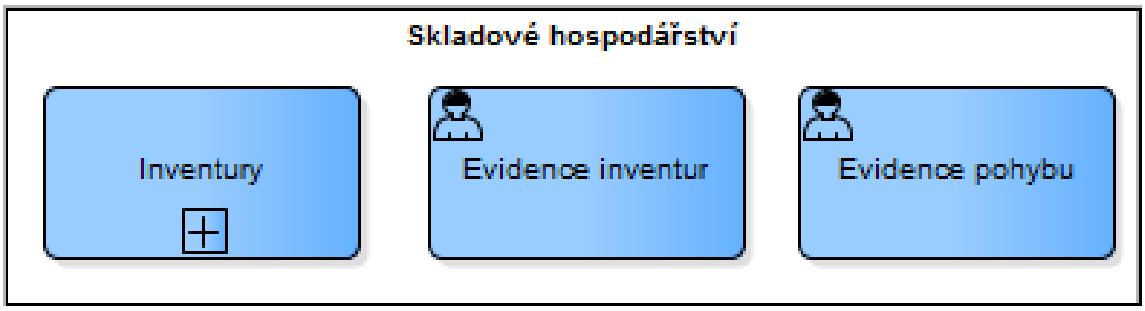
Správa inventur

Procesy ve Správě inventur se zabývají inventurou a příslušnými kroky, které se vytvářejí v rámci daných inventur. Taktéž je důležité, aby se uživatelé

mohli podívat na stav inventur anebo na pohyby, podle kterých se dané inventury budou řešit.

Inventura

Proces zabývající se inventurní činností. Tedy kontrola skutečného stavu zásob s příslušnou agendou, která byla firmou expedována pro daňové účely. V případě nesouladu musí dojít ke kompenzaci či nalezení problému, který nastal.



Obr. 45 první úroveň Správy Inventur Zdroj: vlastní tvorba

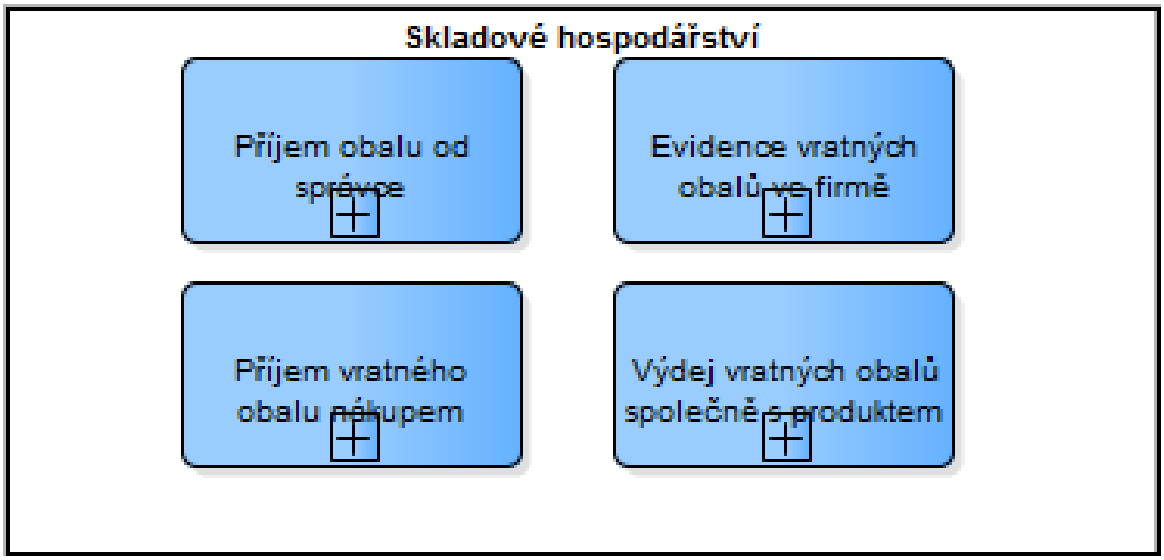
Evidence inventur

Aktivita, ve které se řeší evidence inventur, jejich vytváření, změna a další úkony s inventurou spojené.

Evidence pohybu

Úzce spojená s Evidencí inventur. Při zpracování inventur musí dojít k vytvoření opravných pohybů, které vyrovnají rozdíly mezi skutečným množstvím zásob na skladě a účetním stavem.

Správa obalových kont



Obr. 46 První úroveň Správy Obalových kont Zdroj: vlastní tvorba

Procesy, zabývající se procesy vztahujícími se na vratné a nevratné obaly. V rámci vratných obalů je nutné evidovat jejich stav a pohyb. Mnohdy jsou tyto obaly velmi drahé, a proto je třeba, aby se sledovaly.

Příjem obalu od správce

Aktivita zabývající se přijetím prázdných obalů od správce obalů, zaevidování prázdných obalů do seznamu prázdných obalů a aktualizace obalového konta správce.

Evidence vratných obalů ve firmě

Procesy, zabývající se evidencí vratných obalů. Je potřeba sledovat současný stav uskladněných obalů na skladu. Ve všech vratných obalech je náš kapitál, či kapitál správce, který se o dané obaly stará.

Příjem vratného obalu nákupem

Nákup vratných obalů pro účely firmy. Nakoupené vratné obaly se zaevidují do firemního majetku a zaktualizuje se stav prázdných obalů ve firmě. Dále budou nové obaly fyzicky uskladněny do skladu prázdných obalů nebo se rovnou přesunou do výroby pro plnění produkty.

Výdej vratných obalů společně s produktem

Evidence výdeje vratných obalových jednotek spolu s produktem na objednávku pro určitého zákazníka. Sníží se zásoba vratných obalů a produktu na straně nás jako dodavatele a zvýší se množství vratných obalů na straně zákazníka.

Správa parametrů a číselníků

Obr. 47 První úroveň Správy parametrů a číselníků

Procesy, zabývající se správou číselníků potřebných pro správné fungování systému. Jak je patrné, v **Business Processu** jsou pouze atomické aktivity. Je to z důvodu, že tyto číselníky se pouze plní a tak procesně nejsou nikterak složité.

Pořízení číselníku, skladových dokladů

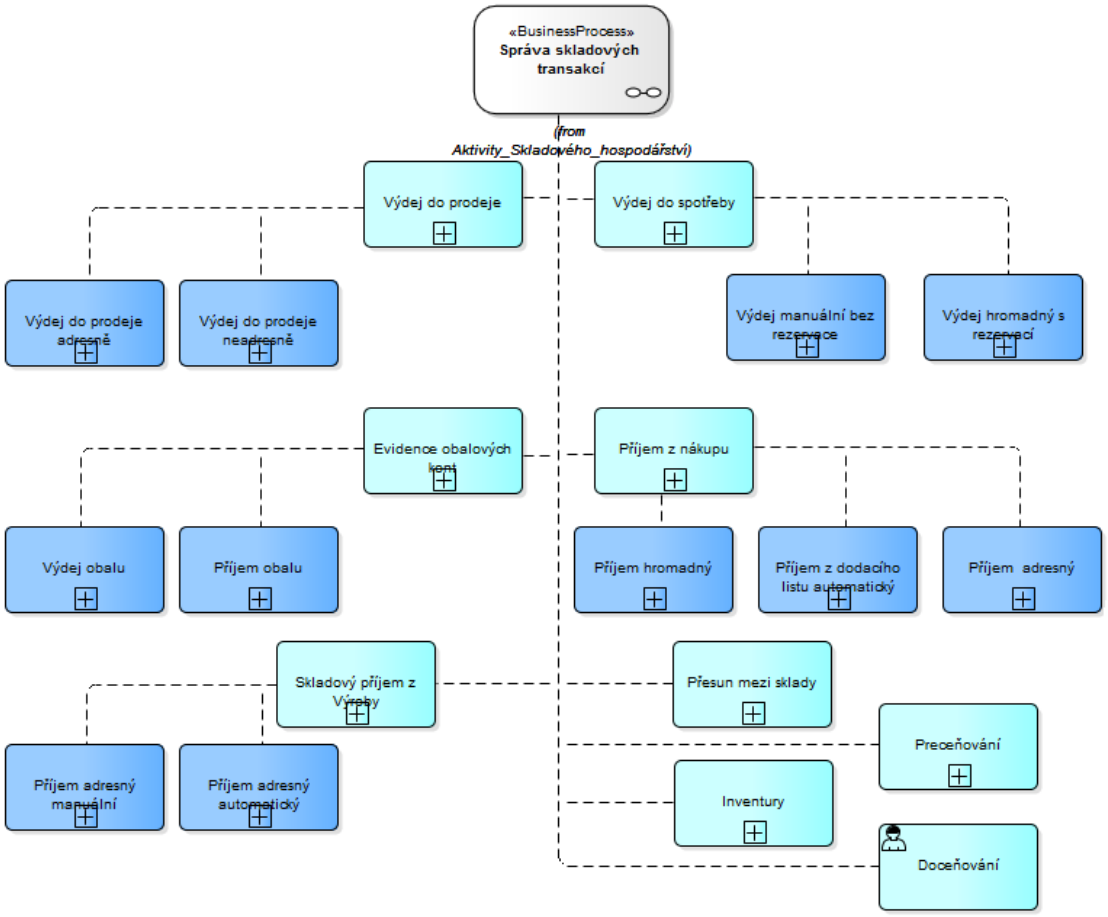
Aktivity charakterizují činnosti, které musí proběhnout na začátku systému, aby systém správně fungoval.

4.5.3 2. Úroveň

Při další úrovni už může nastat menší komplikace s přesnější identifikací procesu. Jelikož nyní jsou dané **sub-procesy** rozvětveny, je nutné, abychom si určili i strom rozvětvení. Podle tohoto stromu se může určit, jaký proces se pojí k jakému business procesu a **sub-procesu**. Dobrým příkladem jsou správy skladových transakcí, jelikož z prvního business procesu nám vzniklo 9 **sub-procesů**. Tyto procesy se ještě níže dělí na nižší **sub-procesy** a je obtížné se vyznat, v jaké větvi daný proces je. Proto byl vytvořen stromový přehled procesů, které rozkrývají základní pohled na členění procesů ve správě skladových transakcí.

2. Úroveň Správy skladových transakcí

Procesy s další úrovní samostatných sub-procesů



Obr. 48 Strom 2. úrovně Správy Skladových transakcí Zdroj: vlastní tvorba

Na obrázku č. 48. vidíme, že se **sub-procesy** v **Business Processu** "Správa skladových transakcí nerozpadají všechny do nižních **sub-procesů**. 5 z těchto procesů se rozpadá do dalších **sub-procesů** a 3 z nich mají definovaný tok už na další úroveň a jedna aktivita Doceňování je definovaná jako atomická aktivita.

Výdej do prodeje

Takto můžeme vidět, že se proces „Výdej do prodeje" ještě dělí na dva možné typy tohoto procesu, a to „Výdej do prodeje adresně", kdy se vyskladňují produkty přímo pro daného zákazníka v rámci jeho objednávky, a na „Výdej do prodeje neadresně" což znamená, že jsou vyskladňovány produkty ze skladu pro určitou objednávku, ale ne pro určitého zákazníka.

Výdej do spotřeby

Tento proces se dále dělí na dva sub-procesy v rámci vychystání. První sub-proces „Výdej manuální bez rezervace“ je pro vychystání menšího počtu produktu, kdy nejsou produkty zarezervovány přesně pro daného zákazníka, ale jsou vychystány na danou objednávku. Druhý **sub-proces** je „Výdej hromadný s rezervací“. To znamená, že jsou všechny rezervované produkty pro příslušného zákazníka vyskladněny a následně zpracovány pro přepravu.

Evidence obalových kont

Zde vidíme rozdělený proces, jak bylo zmiňováno výše v popisu tohoto procesu, na výdej a příjem obalu.

Příjem z nákupu

Tento proces byl dále rozčleněn na 3 možné typy procesu, které mohou nastat. A to „Příjem hromadný“, kdy se přijímá na sklad hromadným příkazem na vícero přijímaných předpisů. „Příjem z dodacího listu automatický“ je proces, kdy se přijímá účetně za pomoci automatického principu, tedy že není zapsán ručně uživatelem do systému, ale vše je přijato automaticky podle dodacího listu, tedy načte se kód dodacího listu a tím jsou načteny i všechny položky z tohoto listu. „Příjem adresný“ je proces, kdy se přijímají produkty, které jsou už určeny pro určitého zákazníka na danou objednávku.

Skladový příjem z výroby

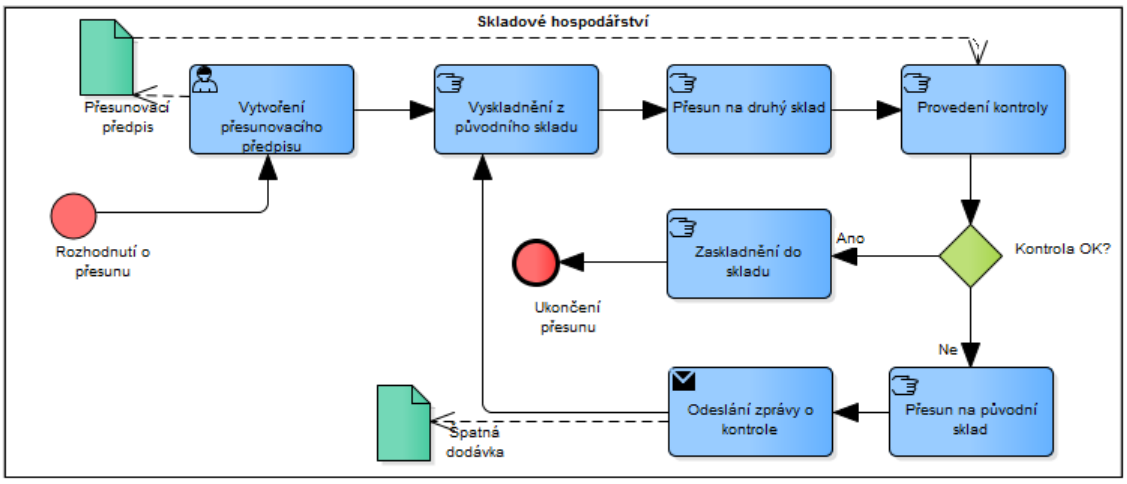
Tento proces je ještě členěn z hlediska manipulace s výrobky po jejich vytvoření. Jedna možnost je, že při každém dokončeném vytvoření výrobku se zaeviduje vytvoření výrobku a odešle na sklad nebo po vyhotovení jedné dávky výrobku je tato zaevidována automaticky na sklad.

Procesy s tokem

Ostatní **sub-procesy** už mají na druhé úrovni tok, který je následně popsán a vymodelován do příslušného diagramu.

Přesun mezi sklady

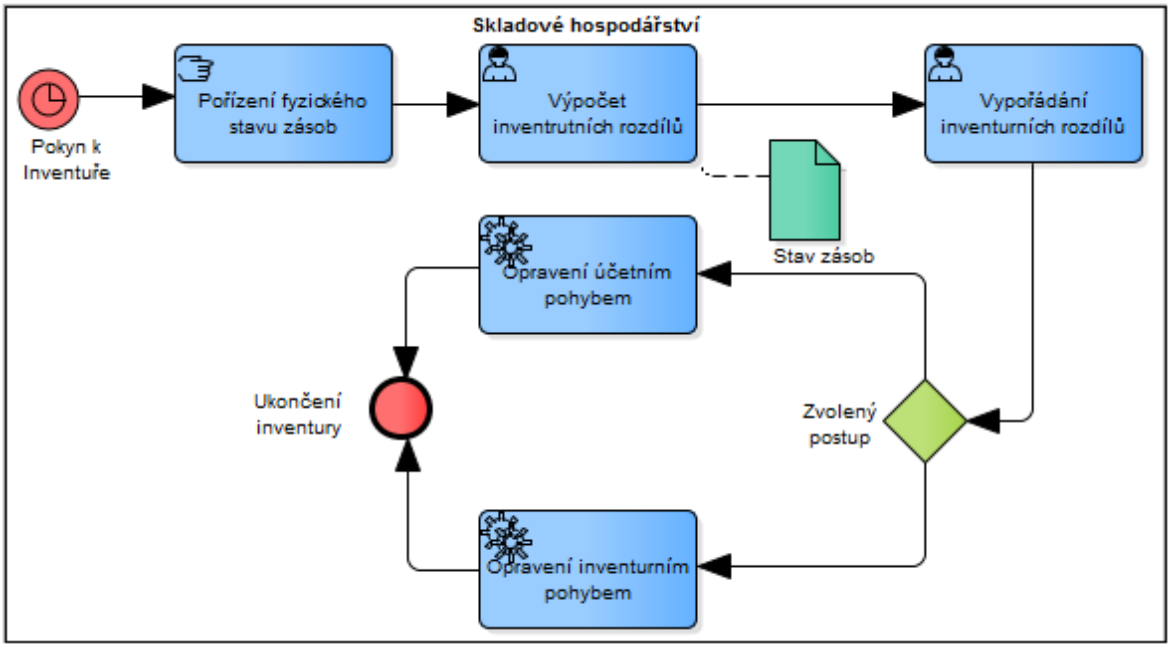
Z diagramu výše je patrné, že pro přesunutí produktu z jednoho umístění na druhé musí být vytvořen přesunovací předpis, který je uživatelsky vytvořen v systému. Podle tohoto předpisu se fyzicky vyskladní dané množství produktu a přesune na druhý sklad, kde má být tento produkt uložen. Při předání na další



Obr. 49 Přesun mezi sklady Zdroj: vlastní tvorba

sklad se provede kontrola správnosti přesunovacího materiálu. Pokud je vše v pořádku, je dané množství produktu uskladněno na nový sklad. V případě nalezení chyby se vše přesune na původní sklad a je odeslána zpráva o nalezení chyby.

Inventury



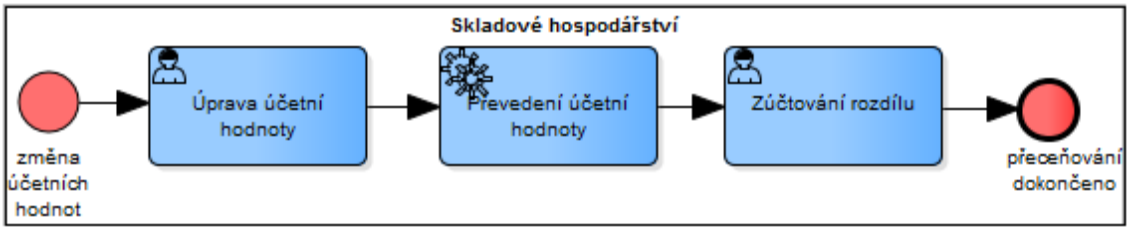
Obr. 50 Inventura Zdroj: vlastní tvorba

Jelikož i tento proces je procesem ve Správě skladových transakcí, je vložena do obou business procesu. Tedy ve Správě skladových transakcí a ve Správě inventur. Ovšem fyzicky se člení pouze do inventur. Zde figuruje jako nalinkovaný proces.

V procesu inventur tak můžeme vidět situaci, že daný proces je spuštěn pomocí časového **Eventu**. To znamená, že daný proces je spuštěn v určitou chvíli.

Poté proběhne zjištění fyzického stavu zásob na skladě. Jednoduše se spočítá, co je ve skladu za zásoby. To vše se porovná se stavem, který je v systému, a vyhodnotí se výsledek. Podle nesrovnalostí se vypořádají inventurní rozdíly. Vznikne tak opravný pohyb, buď Opravením účetním pohybem, nebo Opravením inventurním pohybem.

Přeceňování



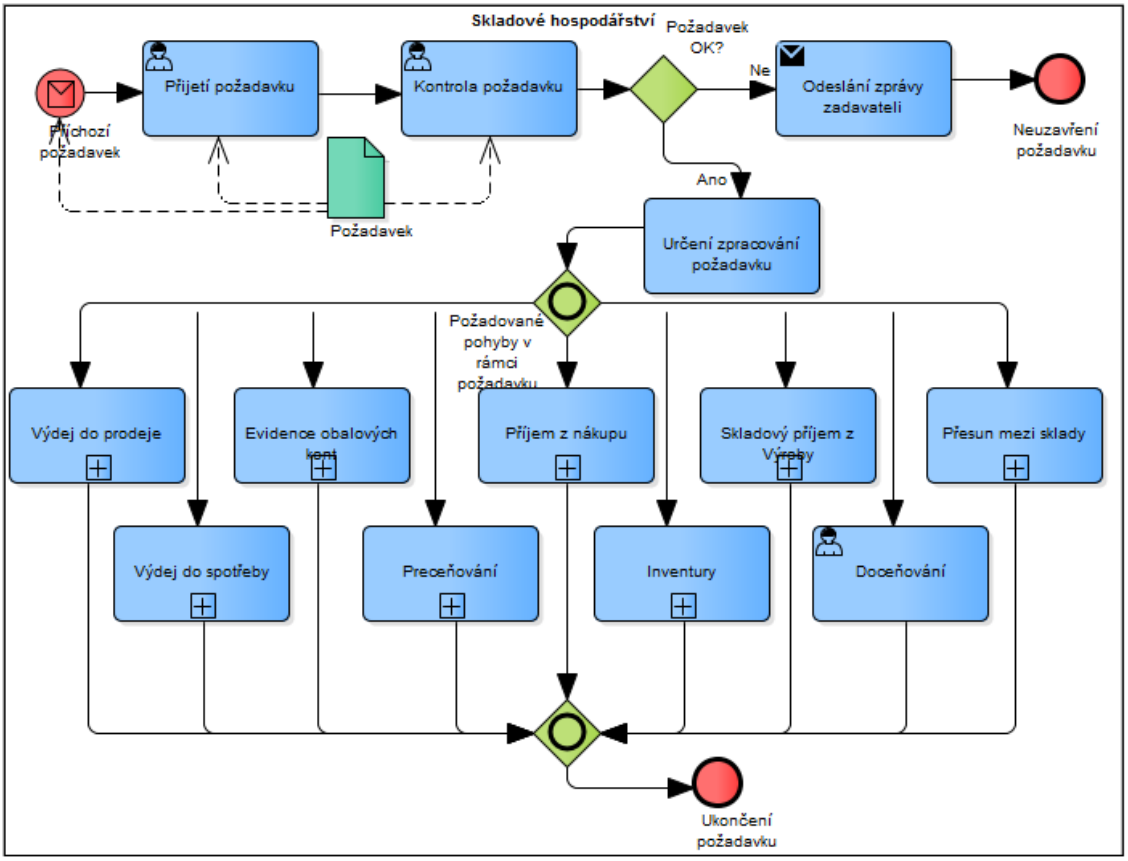
Obr. 51 Přeceňování Zdroj: vlastní tvorba

Při změně účetní hodnoty daných produktů musí dojít i ke změně této hodnoty v systému. Podle nově zadané hodnoty se vypočítají nové hodnoty pro všechny produkty a zúčtuje se rozdíl mezi hodnotou minulou a současnou.

2. Úroveň Správy Zásob

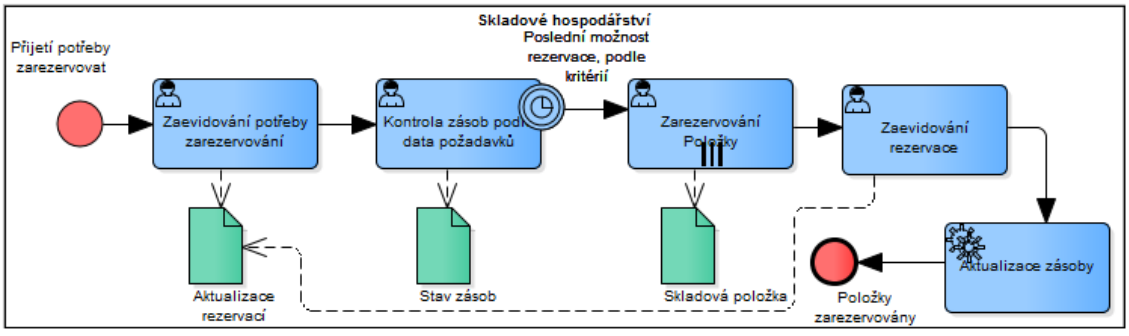
Správa zásob se taktéž vztahuje ke skladovým transakcím, tudíž je vytváření požadavků v rámci manipulace zásob již popsáno ve skladových transakcích a je zmíněno v procesu, jako odkaz na tyto procesy.

Jak je možné vidět z procesu uvedeného níže na obrázku č. 52. celý proces začíná přijetím zprávy o vytvoření požadavku na sklad. Po přijetí požadavku musí proběhnout kontrola požadavku. Pokud je v požadavku popsáno vše, co je potřeba, a splňuje všechny předpoklady, může dojít k určité transakci podle charakteru požadavku nebo může nastat více procesů v rámci jednoho požadavku s různým časovým horizontem. V případě nedostatečného popisu požadavku či chybného popsání požadavku se odesílá zadavateli zpráva o chybném vyplnění požadavku a celý proces je ukončen.



Obr. 52 Druhá úroveň Správy zásob Zdroj: vlastní tvorba

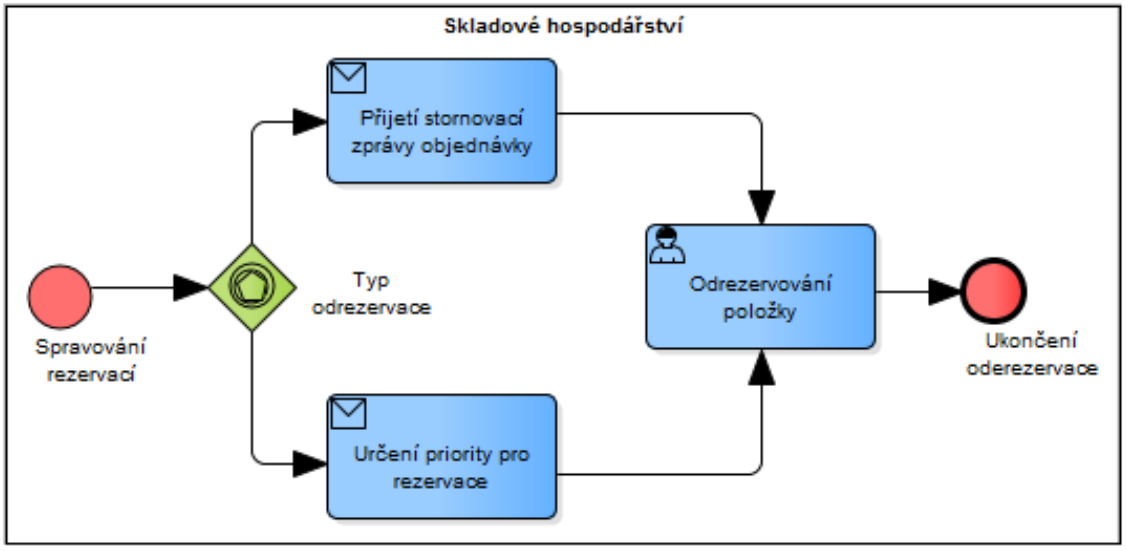
2. úroveň Správy Rezervací



Obr. 53 Rezervace Zdroj: vlastní tvorba

Prvním procesem ve Správě rezervací, je proces Rezervace. Tento proces zajišťuje zarezervování potřebného množství produktu v rámci jedné objednávky pro daného zákazníka. Z diagramu je patrné, že jsou všechny požadavky na rezervaci evidovány do systému a po jejich uložení se kontroluje stav zásob. Podle stavu zásob a zadaných algoritmů se zarezervuje potřebné množství produktu v případě, že jsou všechny potřebné produkty na skladě nebo pokud nejsou v naplánovaném stavu výroby.

Druhý proces ve Správě rezervací je proces Rezervace. Tento proces zajišťuje odrezervování již zarezervovaných produktů, podle druhu požadavku. Mohou nastat dva typy požadavku, díky kterým se zarezervované produkty od-rezervují. V prvním případě se mění stav objednávky. Zákazník požaduje nižší množství produktu, a proto musí dojít alespoň částečně k odrezervování daného produktu, aby byly objednávka a zarezervované produkty pro danou objednávku ve shodě.

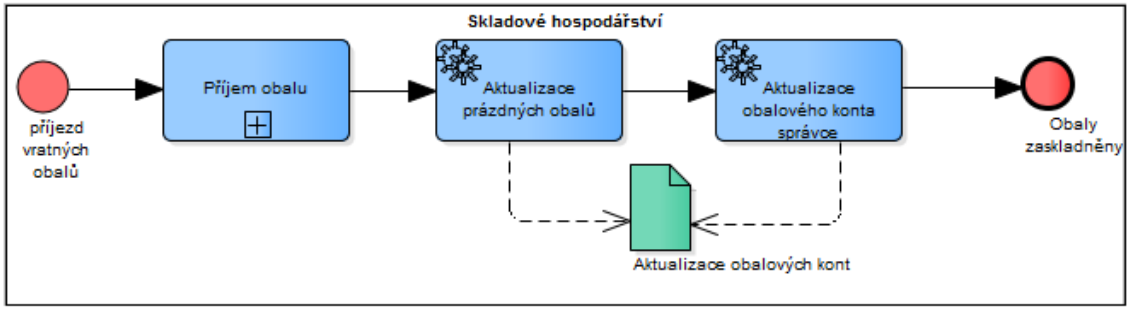


Obr. 54 Od-rezervace Zdroj: vlastní tvorba

2. Úroveň Správy obalových kont

Všechny následující procesy jsou již na této úrovni modelovány do podoby toku. Pouze Evidence vratných obalů ve firmě je hierarchie aktivit.

Příjem obalu od správce



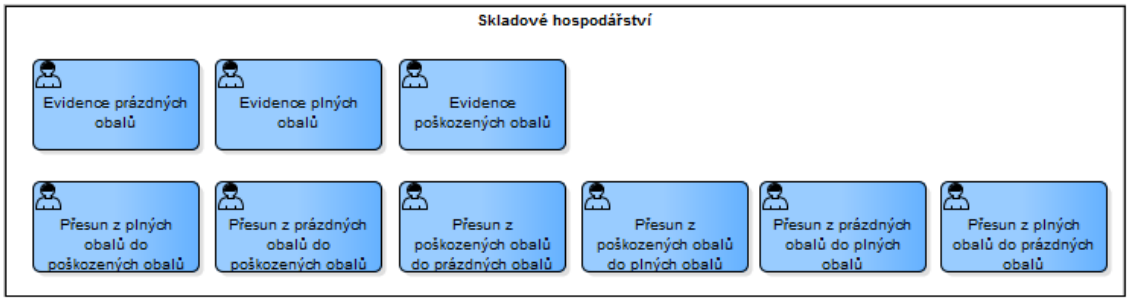
Obrázek 55 Příjem obalu od správce zdroj: vlastní tvorba

Model nám ukazuje, že spouštěcí událostí je příjezd vratných obalů do firmy. K tomu slouží proces příjmu obalů na sklad. Tento stav může být vymodelován do současného stavu, takže nám vznikne větší diagram, ovšem tento

postup se používá i v jiných diagramech, a proto je jeho vymodelování přímo na vyšší úroveň nežádoucí. Po přijetí obalu na sklad dojde k aktualizaci prázdných obalů. Dojde tedy ke zvýšení stavu zásoby na skladu, a nové obaly jsou tak připraveny pro zabalení výrobků. Taktéž dochází ke zvýšení konta správce obalů. Zvýší se zásoba jeho obalů v našich stavech obalového konta.

Evidence vratných obalů ve firmě

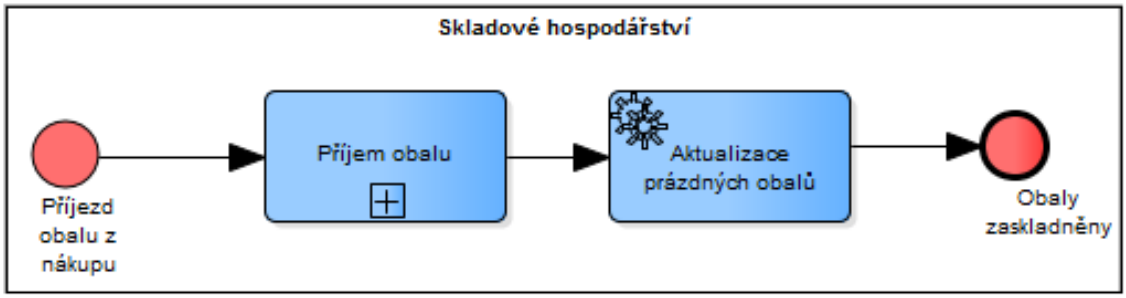
Tento proces je převážně vytvořen pro evidenci obalu a jeho přesun z jednoho stavu do druhého. V rámci jedné firmy je nutné evidovat nejen počet vratných obalů, ale taktéž jejich vnitřní stavy. Pokud je obal připraven pro balení, je označen jako prázdný obal. Obal, který je již použit, a je v něm zabalen produkt, je označen jako plný obal. Obal, který již není možné použít z důvodu znehodnocení, je označen jako poškozený obal. Takto se v rámci firmy organizují obaly a jednotlivé přesuny značí právě přechody mezi stavy.



Obr. 55 Evidence vratných obalů ve firmě Zdroj: vlastní tvorba

Příjem vratného obalu nákupem

Text je stejný jako u procesu „Příjem obalu od správce“. Poslední věta u popisu ze zmiňovaného procesu zde není, jelikož popisuje poslední aktivitu, která zde není. Model nám ukazuje, že spouštěcí událostí je příjezd vratných obalů do

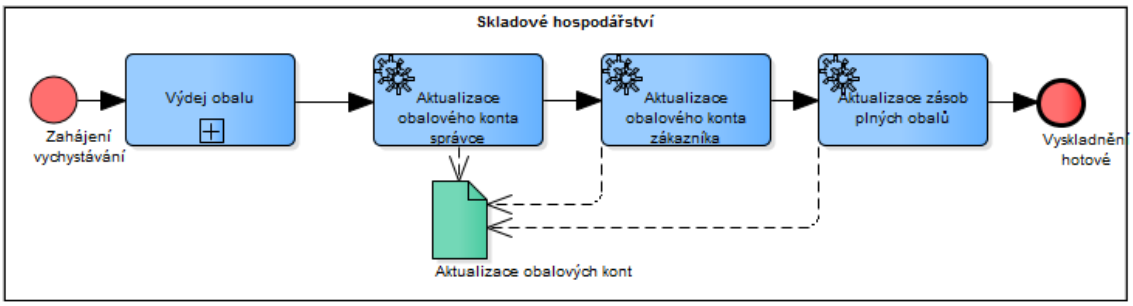


Obr. 56 Příjem vratného obalu nákupem Zdroj: vlastní tvorba

firmy. Na to je proces příjmu obalů na sklad. Tento stav může být vymodelován do současného stavu, takže nám vznikne větší diagram, ovšem tento postup se používá i v jiných diagramech a proto je jeho vymodelování přímo na vyšší úroveň nežádoucí. Po přijetí obalu na sklad dojde k aktualizaci prázdných obalů. Dojde tedy ke zvýšení stavu zásoby na skladu a nové obaly jsou tak připraveny pro zabalení výrobků.

Výdej vratných obalů společně s produktem

Tento proces řeší vyskladnění vratného obalu pro zabalení produktu a jeho vyskladnění v rámci produktu. Při vyskladnění produktu zabaleného do vratného obalu dochází k aktualizaci třech kont, a to obalového konta správce a zákazníka. Tyto dvě konta jsou zvětšena právě o počet obalů, které jsou vyskladněny v rámci jejich objednávky. Současně dochází k aktualizaci našeho konta plných obalů. Toto konto se sníží o počet vratných obalů, které byly vyexpedovány pro daného zákazníka. Zvýšení obalového konta znamená fakt, že v případě vybalení produktu z obalu je správce zodpovědný za navrácení obalu do naší firmy.

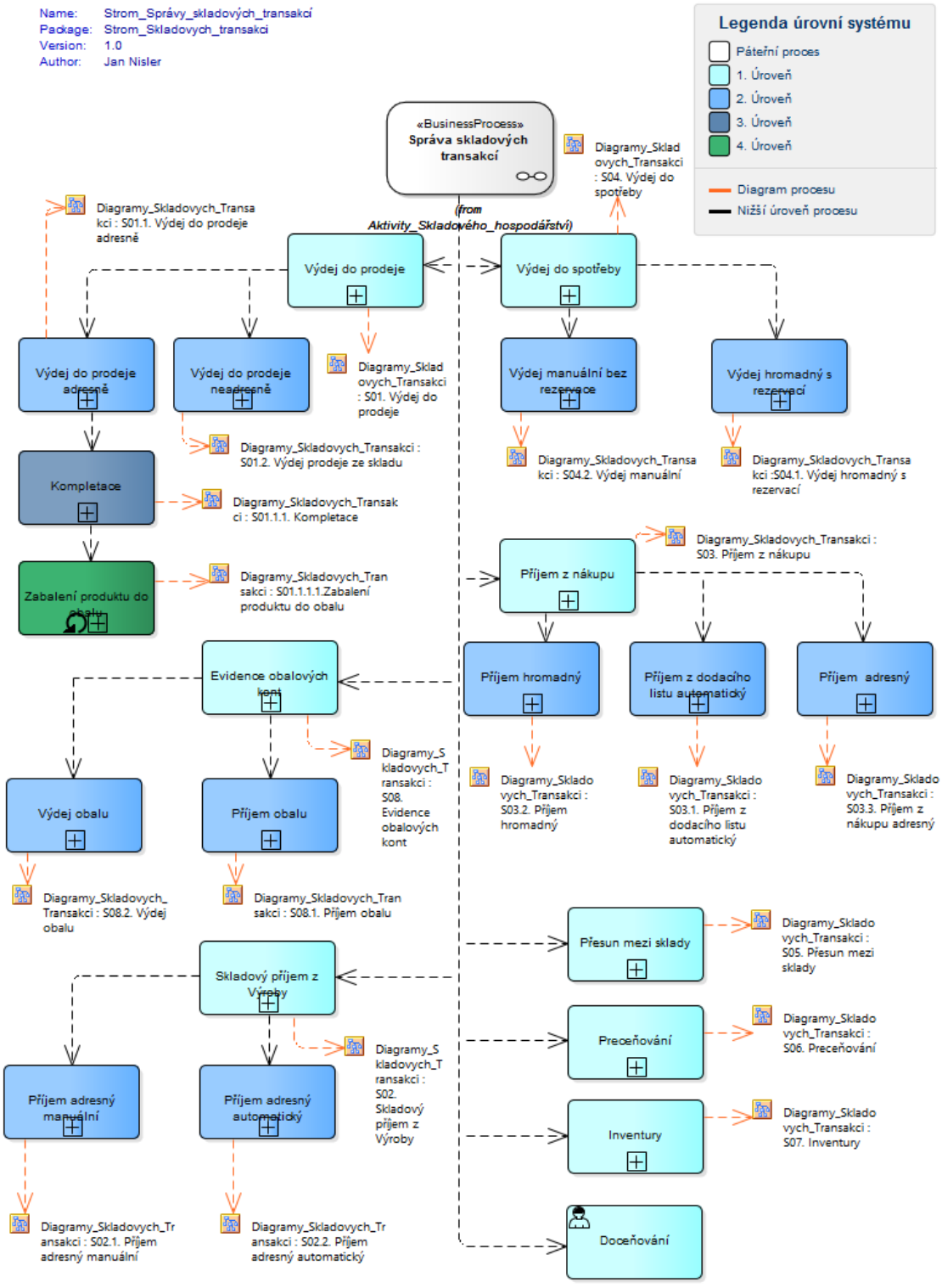


Obr. 57 Výdej vratných obalů společně s produktem Zdroj: vlastní tvorba

4.5.4 3. Úroveň a nižší

Mohlo by se zdát, že vypracování „pouze“ skladového hospodářství v základním systému je velmi nepřehledné a uživatel nemá šanci se v něm vyznat. Ovšem na to, abychom se vyznali v takovéto mapě, nám poslouží základní procesní strom. Tuto funkci bohužel Enterprise Architect neumí, ovšem jeho vypracování je více méně velmi jednoduché a při praxi v nástroji i časově velmi úsporné.

Správa skladových transakcí



Obr. 58 Strom Správy skladových transakcí Zdroj: vlastní tvorba

Z níže vloženého diagramu můžeme jasně vidět jednotlivé úrovně, které v systému zatím máme. Pro lepší přehlednost a srozumitelnost jsou jednotlivé úrovně odlišeny barevně a je zde vytvořená legenda, ve které může čtenář nalézt i

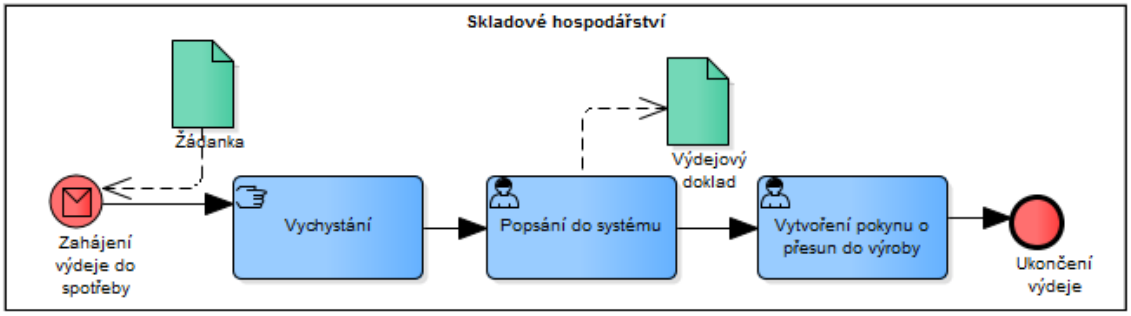
informaci, jaké vazby se v diagramu nacházejí. Tento strom nám ukazuje jednotlivá řešení jednoho páteřního procesu. Kdybychom chtěli všechny páteřní diagramy popsat do jednoho diagramu, vznikl by nám veliký formát, kterých bychom na klasický list nevytiskli. Proto jsou rozlišeny do jednotlivých páteřních procesů, i když musíme použít duplicitu, kdy se v procesech používají stejné procesy či **Tasky**. Směr vazby vždy určuje vztah od předka k potomku, tedy od vyšší úrovně k nižší. Zde už tedy můžeme identifikovat obě úrovně, které byly dosud popsány. Procesy s tokem na 3 úrovně či níže můžeme taktéž vidět na diagramu níže. Oranžová vazba značí přechod procesu na tok procesu. Nyní je tedy nutné určit zbývající toky procesů, které nebyly upřesněny v druhém kroku.

Výdej do spotřeby

Tento proces, jak je uvedeno výše, je dělen na několik typů procesu. Tyto procesy jsou následující:

Výdej manuální bez rezervace

Tento proces pojednává o výdeji produktu do výroby, kde je následně zpracován do další podoby. Zpráva o vychystání se podává v rámci žádanky, která je zpracována a na základě informací v ni vložených se vyskladní požadované množství produktu ze skladu do výroby. Vychystání se zapíše do systému a vytvoří se pokyn o přesunu produktu do výroby.

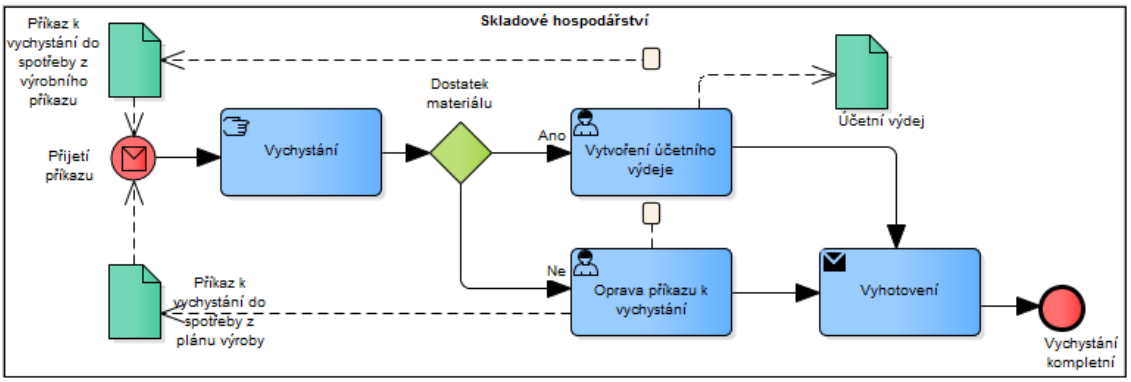


Obr. 59 Výdej manuální bez rezervace zdroj: vlastní tvorba

Výdej hromadný s rezervací

Tento proces pojednává o hromadném vyskladnění produktu ze skladových umístění do výroby podle dvou příkazů. První příkaz je tvořen výrobním příkazem, ve kterém je specifikováno, kolik množství produktu je zapotřebí pro vyhotovení produktu. V závislosti na počtu je vyskladněno požadované množství produktu ze skladu. Druhý příkaz je příkaz k vychystání do spotřeby z plánu výroby. Jak z názvu

vyplývá, plánovaná výroba je druh výroby, jejíž jednotlivé provádění závisí na plánu, který byl definován v určitém časovém okamžiku a je platný pro daný časový interval. Díky tomuto plánu je jasné, co kdy bude vyráběno, a tak i pro skladové hospodářství je jasné, kdy se jaké produkty musí pro plán vyskladnit. Když přijde takovýto příkaz na sklad, vychystá se požadované množství produktu. V případě dostatku materiálu se dané množství účetně vydá a předá na výrobu. Pokud není dostatek materiálu na skladu z důvodu včasného nezarezervování produktu, je množství produktu opraveno o skutečné množství vychystávaného produktu a předáno k vychystání.



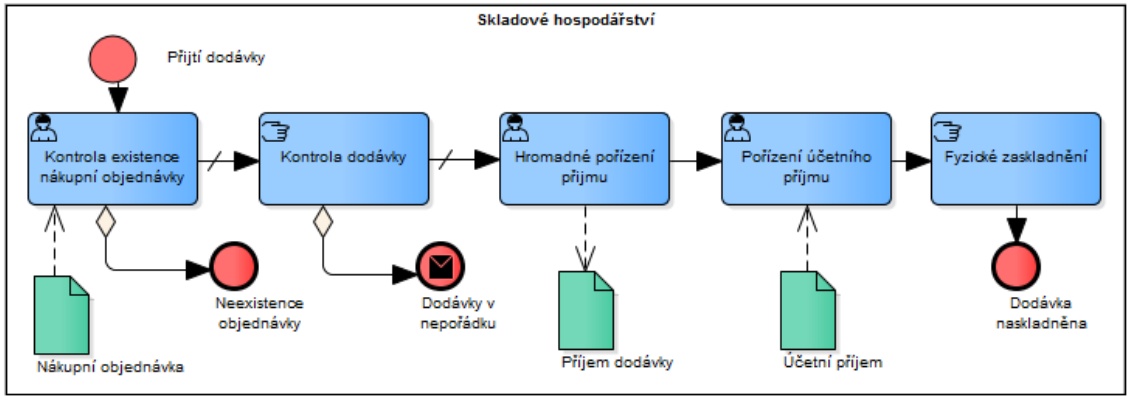
Obr. 60 Výdej hromadný s rezervací Zdroj: vlastní tvorba

Příjem z nákupu

Tento proces může být vytvářen třemi variantami, které jsou popsány níže.

Příjem hromadný

Tento proces přijímá na sklad produkty hromadně. Při přijetí dodávek dojde ke kontrole s nákupní objednávkou, která je zaevidovaná v systému. Řeší se pouze dodavatel. Při kontrole dodávky z důvodu poškození se může označit přijatá dodávka jako v pořádku a je poté hromadně pořízena do systému jako přijetí na sklad. V rámci tohoto příjmu proběhne účetní příjem na sklad. To znamená, že se

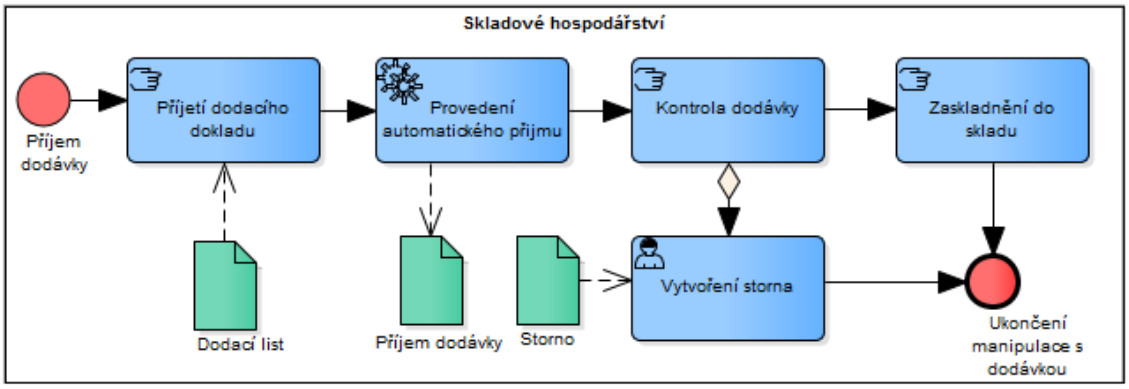


Obr. 61 Příjem hromadný Zdroj: vlastní tvorba

cena kapitálu na skladě v rámci zásob zvedne. Po těchto krocích dochází k fyzickému zaskladnění na sklad podle předem zadaného stylu zaskladnění.

Příjem z dodacího listu automatický

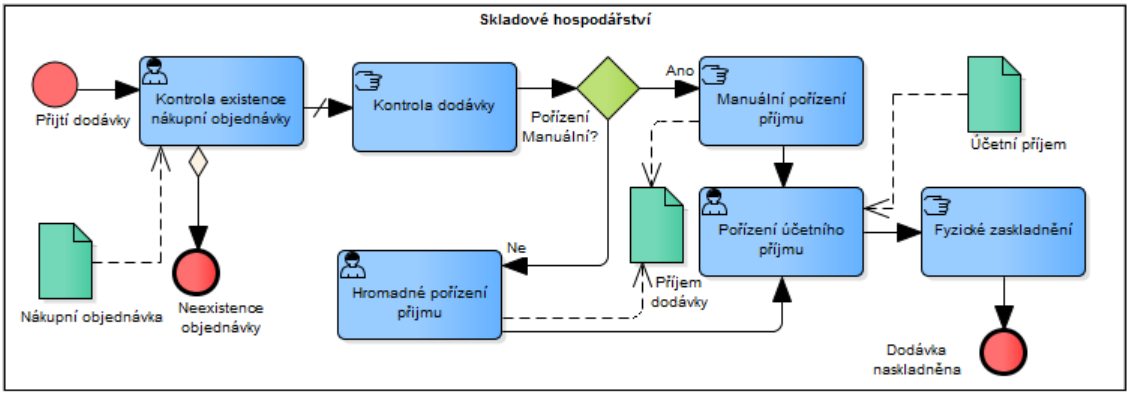
Tento proces ukazuje automatické přijetí produktu na sklad v závislosti na dodacích listech. Při přijetí dodávky nám přepravce či dodavatel předají dodací doklad. Tento doklad se načte do systému a v jeho rámci se provede automatický příjem na sklad. Poté teprve dochází k fyzické kontrole dodávky. V případě bezchybnosti dochází k zaskladnění všech produktů na sklad. V případě nalezení chyby se vytvoří storno na zadané produkty.



Obr. 62 Příjem z dodacího listu automatický Zdroj: vlastní tvorba

Příjem adresný

Tento proces se zabývá přijetím produktu adresně, a to buď hromadně, nebo jednotlivě. Při přijetí dodávky na sklad dochází k její kontrole v systému. Po této kontrole následuje kontrola dodávky, kde se v případě kladného výsledku dodají produkty na sklad buď manuálně po jednom, nebo hromadně. Po těchto aktivitách dochází ještě k účetnímu příjmu na sklad. Po těchto systémových přijetích dochází k fyzickému příjmu na sklad.



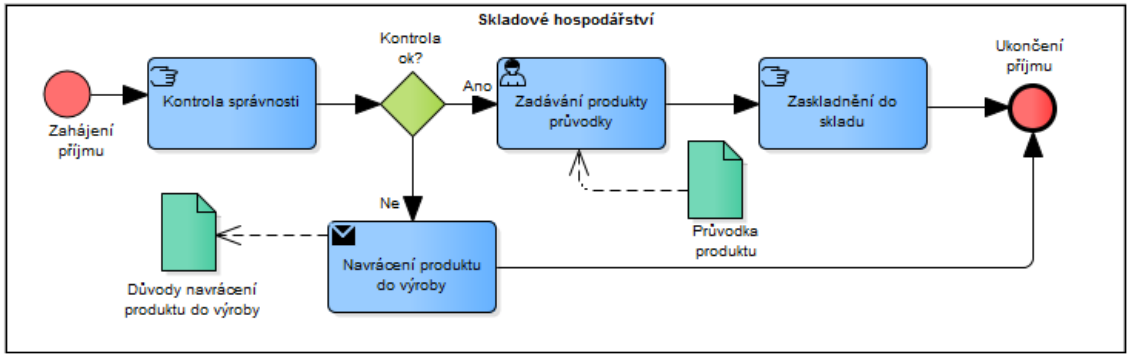
Obr. 63 Příjem adresný Zdroj: vlastní tvorba

Skladový příjem z výroby

Tento proces se zabývá, jak bylo zmíněno výše, přijetím produktu z výroby. Daný proces se dělí na dva typy přijetí na sklad.

Příjem adresný manuální

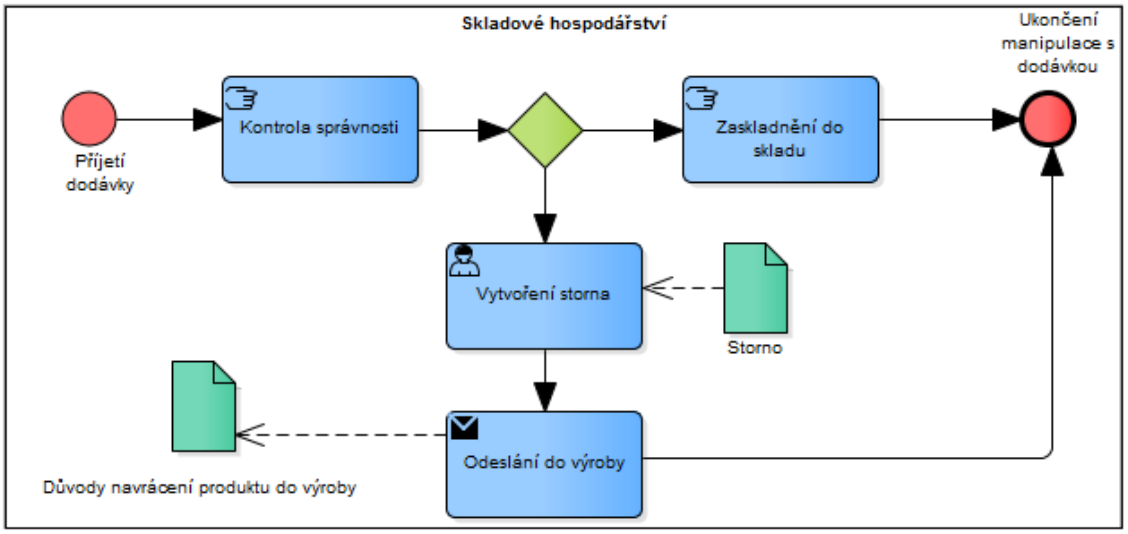
Tento proces je zaměřen na příjem produktu z výroby. V době přijímání produktu dochází ke kontrole produktu. Pokud je kontrola v pořádku, zadají se produkty průvodky do systému a fyzicky se zaskladní do skladu. V případě nalezení chyby při kontrole produktu dochází k navrácení produktu do výroby, kde je opraven, nebo použit na jiné zpracování výroby.



Obr. 64 Příjem adresný manuální Zdroj: vlastní tvorba

Příjem adresný automatický

Tento proces začíná přijetím dodávky do systému. Při tom se zkontrolují produkty. Pokud jsou bezchybné, zaskladní se produkty na sklad. V případě nalezení chyby se vytvoří storna a produkty se odešlou zpět do výroby.



Obr. 65 Příjem adresný automatický Zdroj: vlastní tvorba

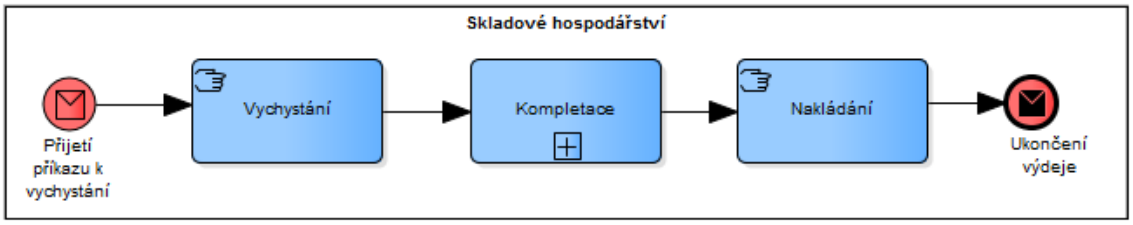
Výdej do prodeje

Jako poslední proces, který je zmíněn a rozložen do nižších podúrovní, je proces Výdej do prodeje. Tento proces je rozčleněn jako jediný do 5 úrovní, které nyní budou popsány.

Samotný proces je ve dvou variantách, které jsou spuštěny v závislosti na druhu objednávky.

Výdej do prodeje adresně

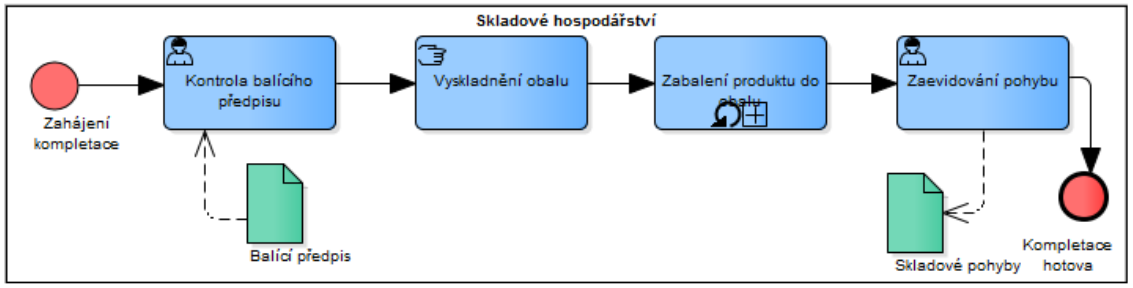
Proces „Výdej do prodeje adresně“ je zahájen přijetím příkazu k vychystání. Na základě informací uložených v tomto příkazu jsou vyskladněny všechny potřebné produkty pro vyhotovení celé objednávky. Při vychystání všech produktů dojde k jejich kompletaci pomocí pravidel stanovených pro určenou velikost přepravovací palety. Po kompletaci dojde k naložení objednávky na přepravní prostředek a ukončí se samotný výdej.



Obr. 66 Výdej do prodeje adresně Zdroj: vlastní tvorba

Kompletace

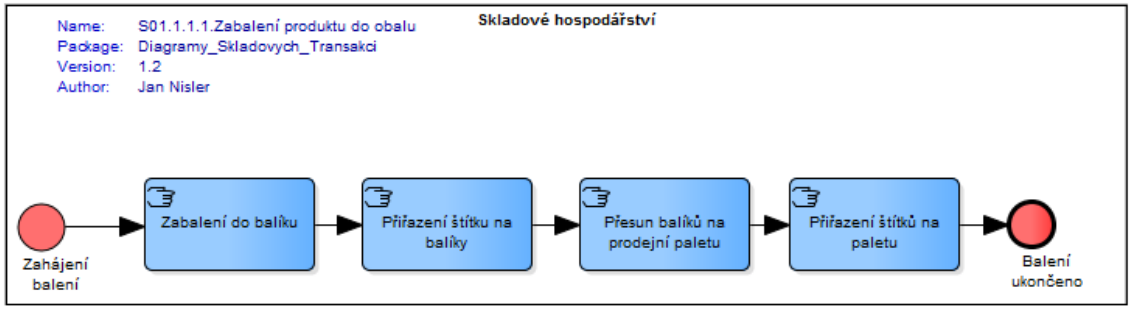
Tento proces je zahájen při vyskladnění všech komponent na objednávku. V prvé řadě je obsluhou zkontrolován balící předpis. Tento předpis je definován zákazníkem a je nutné, aby jej dodavatelé dodržovali. Při identifikaci balícího předpisu jsou na jeho základě vyskladněny příslušné obaly. Zde se poukazuje na fakt, že je produkt zabalen i na prodejní paletu, která má být součástí balícího předpisu. Po zabalení produktu se zaeviduje pohyb a kompletace je tímto ukončena.



Obr. 67 Kompletace Zdroj: vlastní tvorba

Zabalení produktu do obalu

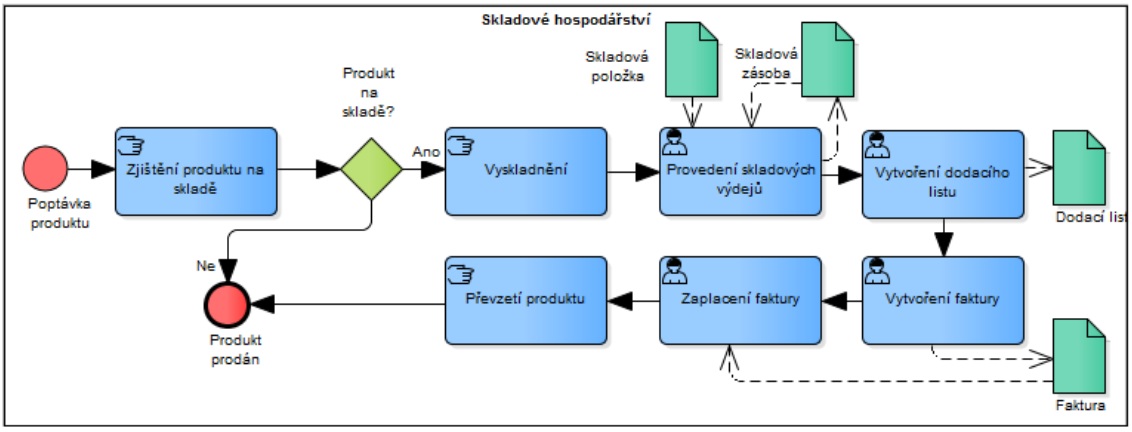
Produkt se podle balícího předpisu a podle definované palety zabalí na prodejní paletu dle příslušného algoritmu. Tento algoritmus se snaží vypočítat všechny možné kombinace uskladnění daného produktu na danou paletu. Na základě rozměrů produktu a palety a maximální výšky palety, která je stanovena, vypočítá optimální rozložení produktu. U menších produktů dochází ještě k zabalení produktu do balíků, které jsou ukládány na prodejní palety. Před samotným zabalením na paletu či do balíku musí být příslušný balící objekt identifikován pomocí štítku. Tento štítek může být tvořen pomocí identifikačních popisků, či 2D kódu. Takto identifikované balíky jsou zabaleny na paletu a i samotná paleta je označena tímto identifikačním štítkem. V tomto okamžiku je balení dokončeno.



Obr. 68 Zabalení produktu do obalu Zdroj: vlastní tvorba

Výdej do prodeje neadresně

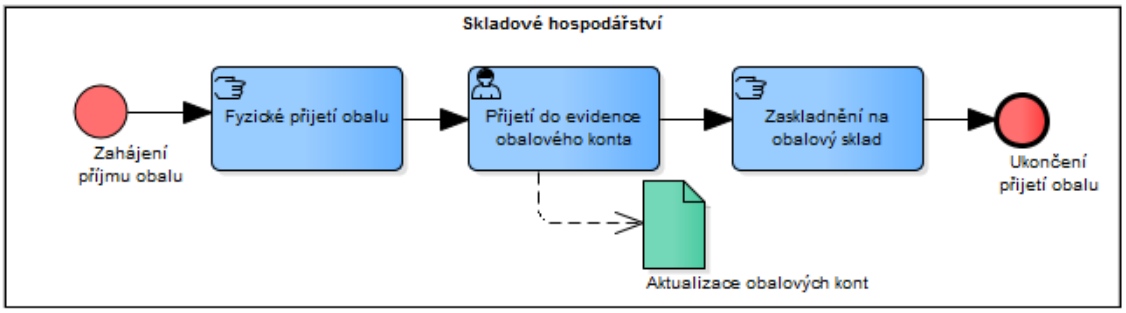
Tento proces vyskladnění je určen pro skladový prodej. Nejsou zde tedy předem dané objednávky, pro které jsou vychystány produkty, ale vychystává se v rámci prodeje, který může být realizovaný přímo na skladě. Při příchodu do skladu (skladového obchodu) si zákazník vybere produkt a v rámci jeho poptávky se zjišťuje, zdali je požadovaný produkt na skladě. Pokud je produkt na skladě, dojde k jeho vyskladnění a provedení skladového výdeje. Poté je vytvořen dodací list a vystavena faktura, kterou má zákazník uhradit. Po zaplacení dochází k předání produktu zákazníkovi a končí celý proces Výdej do prodeje neadresně. Pokud není produkt na skladě, zákazník může odejít, či vybrat jiný.



Obr. 69 Výdej do prodeje neadresně Zdroj: vlastní tvorba

Příjem obalu

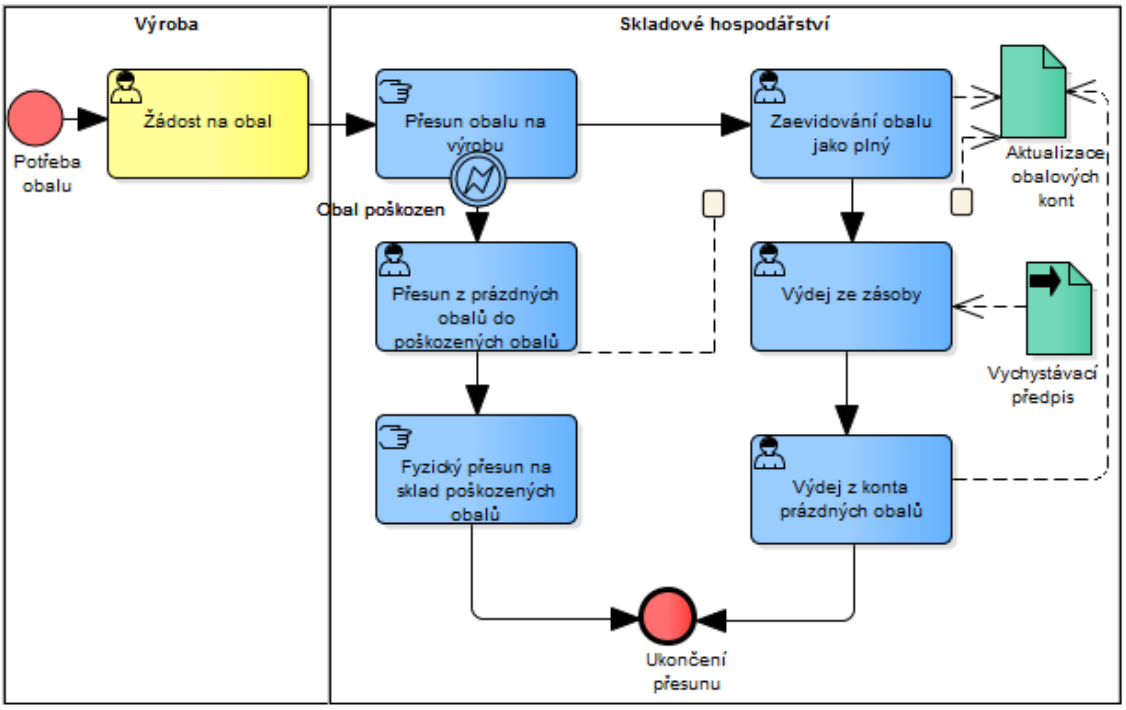
Proces přijetí obalu začíná přijetím obalu do skladových prostor. Fyzicky se příjme dodávka obalu a zaeviduje se přijetí obalu do evidence obalového konta. Poté se veškeré přijaté obaly zaskladní do skladu obalů. Tímto je proces přijetí obalu ukončen.



Obr. 70 Příjem obalu Zdroj: vlastní tvorba

Výdej obalu

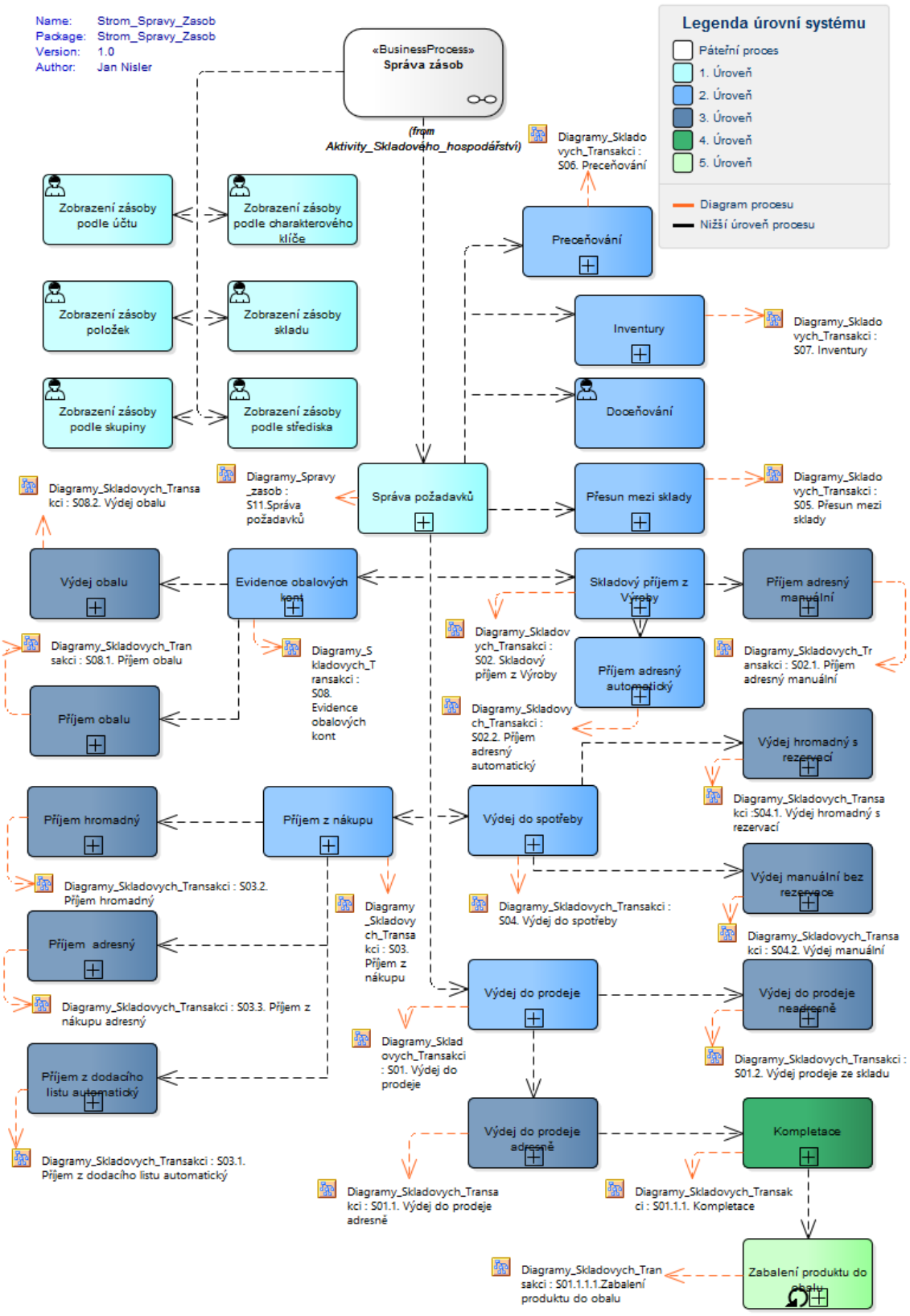
Tento proces je záměrně trochu jiný, a to za účelem znázornění návaznosti výroby na skladové hospodářství. Tyto dvě oblasti jsou spolu úzce propojeny a mnoho požadavků pochází právě z výroby. Tento proces ukazuje výdej obalu do spotřeby v případě, že výroba požádá o vyskladnění vratného obalu pro vyráběný produkt. Při vyskladnění produktu může obsluha objevit, že přepravovaný obal je poškozen a tak je přesunut na sklad poškozených obalů, jak fyzicky tak evidenčně. Pokud je obal v pořádku, je přesunut do výroby. Daný obal je evidenčně přesunut na obalové konto plných obalů. Zaeviduje se výdej ze zásoby obalů a vytvoří se výdej z konta prázdných obalů. Tím je proces ukončen.



Obr. 71 Výdej obalu Zdroj: vlastní tvorba

Správa zásob

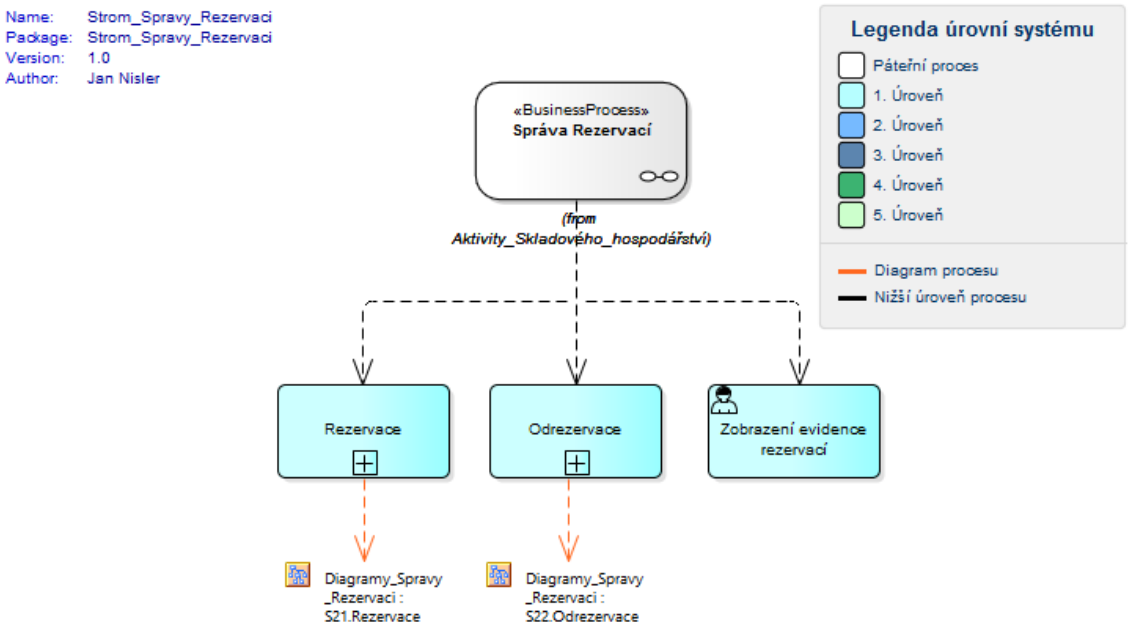
Tento páteřní proces je v celém stromu vyobrazen níže. Je patrné, že jsou zde obsaženy procesy ze Správy skladových transakcí. Zde jsou ovšem tyty procesy na nižší úrovni, a proto mají jinou barvu nežli v předchozím stromu. Je ovšem dobře čitelné, jak jsou tyto procesy navzájem propojené, a jak a z jakého místa mohou být jednotlivé procesy volány.



Obr. 72 Strom Správy zásob Zdroj: vlastní tvorba

Správa rezervací

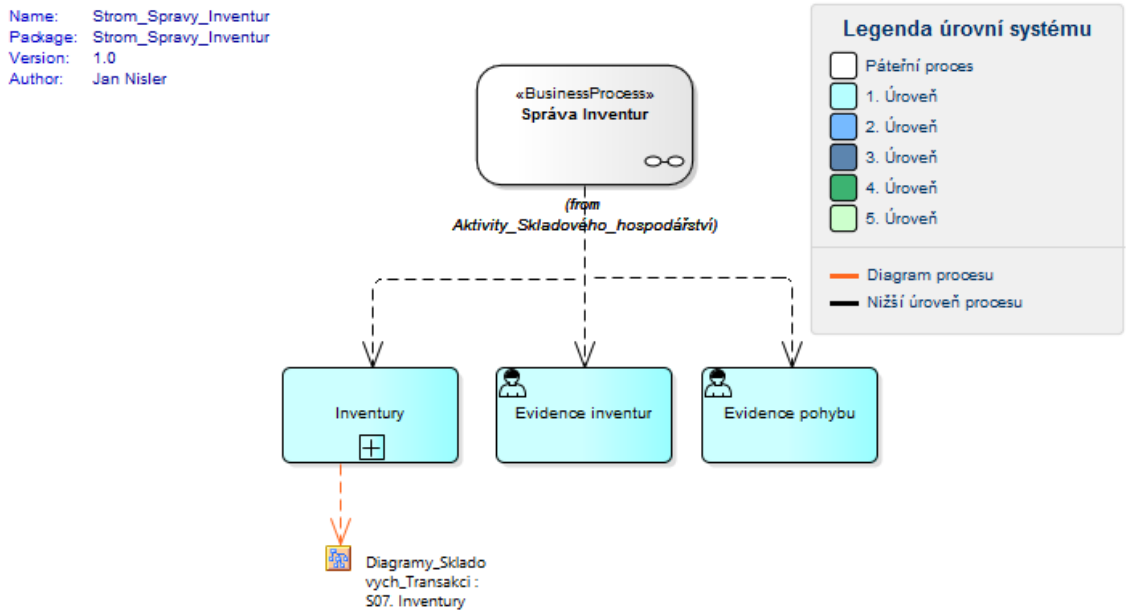
Tento páteřní proces není procesně bohatý, a proto končí na 1. úrovni. Tyto procesy byly popsány výše a tvoří ucelenou oblast.



Obr. 73 Strom Správy rezervací Zdroj: vlastní tvorba

Správa inventur

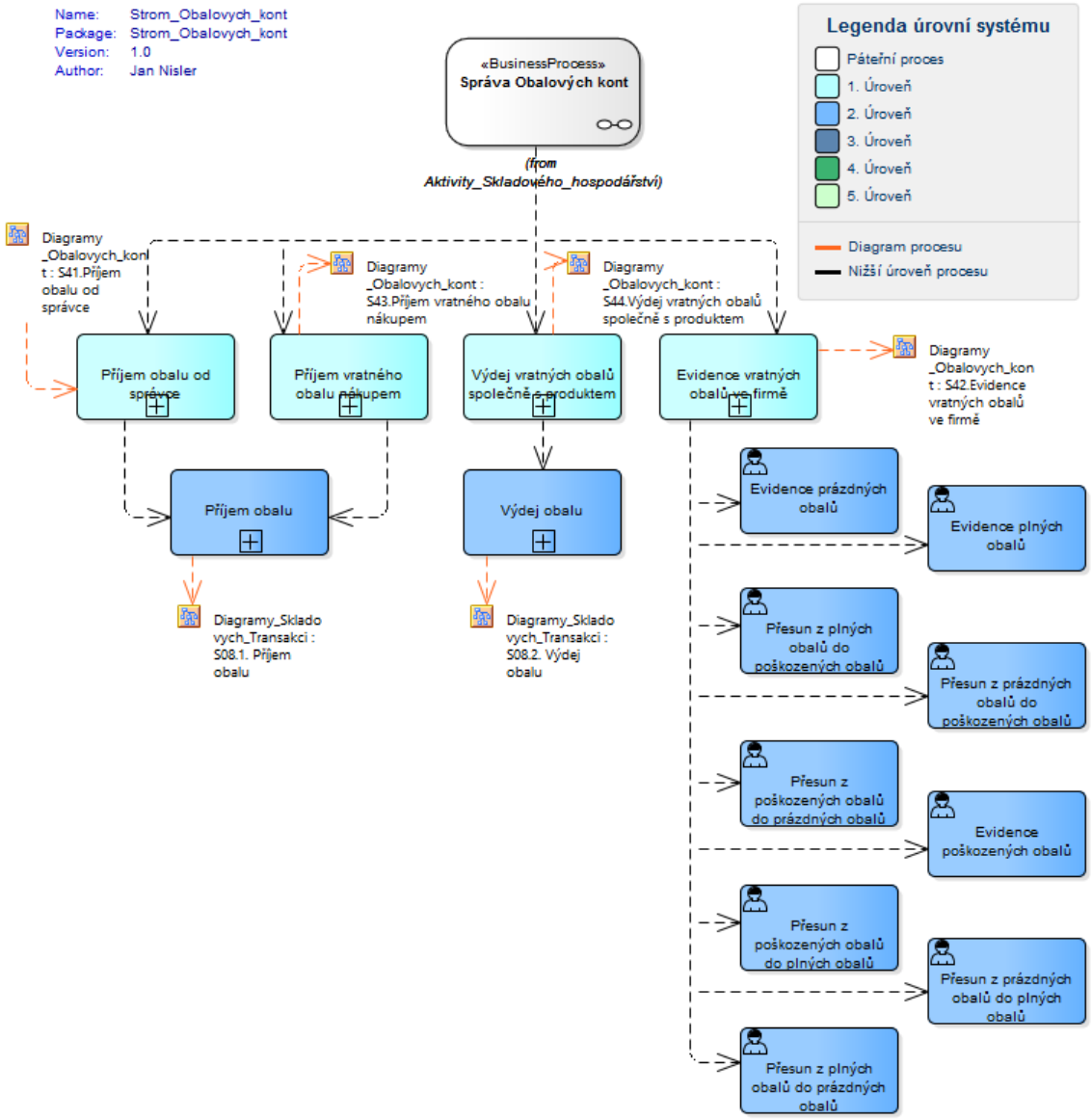
Tento páteřní proces je jako předchozí páteřní proces velmi chudý na procesní mapu. I když se zde objevuje proces, který byl už obsažen u dvou dalších diagramů. Tento proces „Inventury“ je již obsažen ve Správě zásob a ve Správě skladových transakcí.



Obr. 74 Strom Správy Inventur Zdroj: vlastní tvorba

Správa Obalových kont

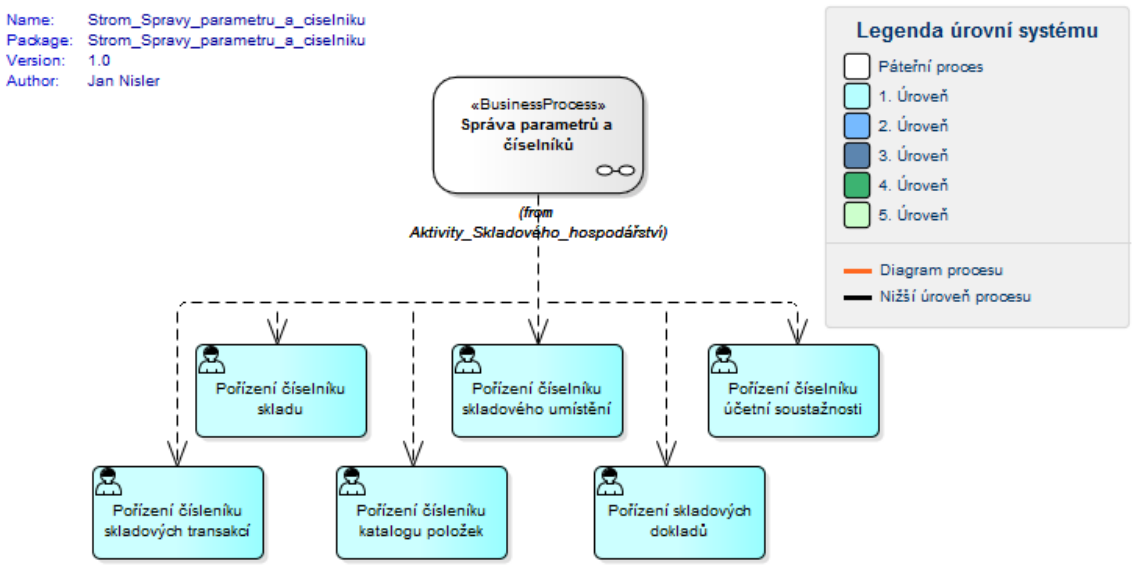
Tento páteřní proces je členěn na 2. úroveň procesu, která je popsána výše a celý strom je vyobrazen na obrázku níže. Sub-procesy Příjem obalu a Výdej obalu jsou již obsaženy v popisu procesu výše v předchozích odstavcích.



Obr. 75 Strom Správy obalových kont Zdroj: vlastní tvorba

Správa parametrů a číselníků

Tento páteřní proces je pouze proces atomický. V tomto procesu se nastavují parametry a číselníky, a proto jsou zde pouze **User Tasky**, které se v systému promítnou jako use case (typové úlohy). I když se zdá, že je tento proces zbytečný, je jedním z nejdůležitějších procesů. Bez tohoto procesu by nemohl fungovat celý systém, a proto je na jeho popsání kladen důraz.



Obr. 76 Strom Správy parametrů a číselníků Zdroj: vlastní tvorba

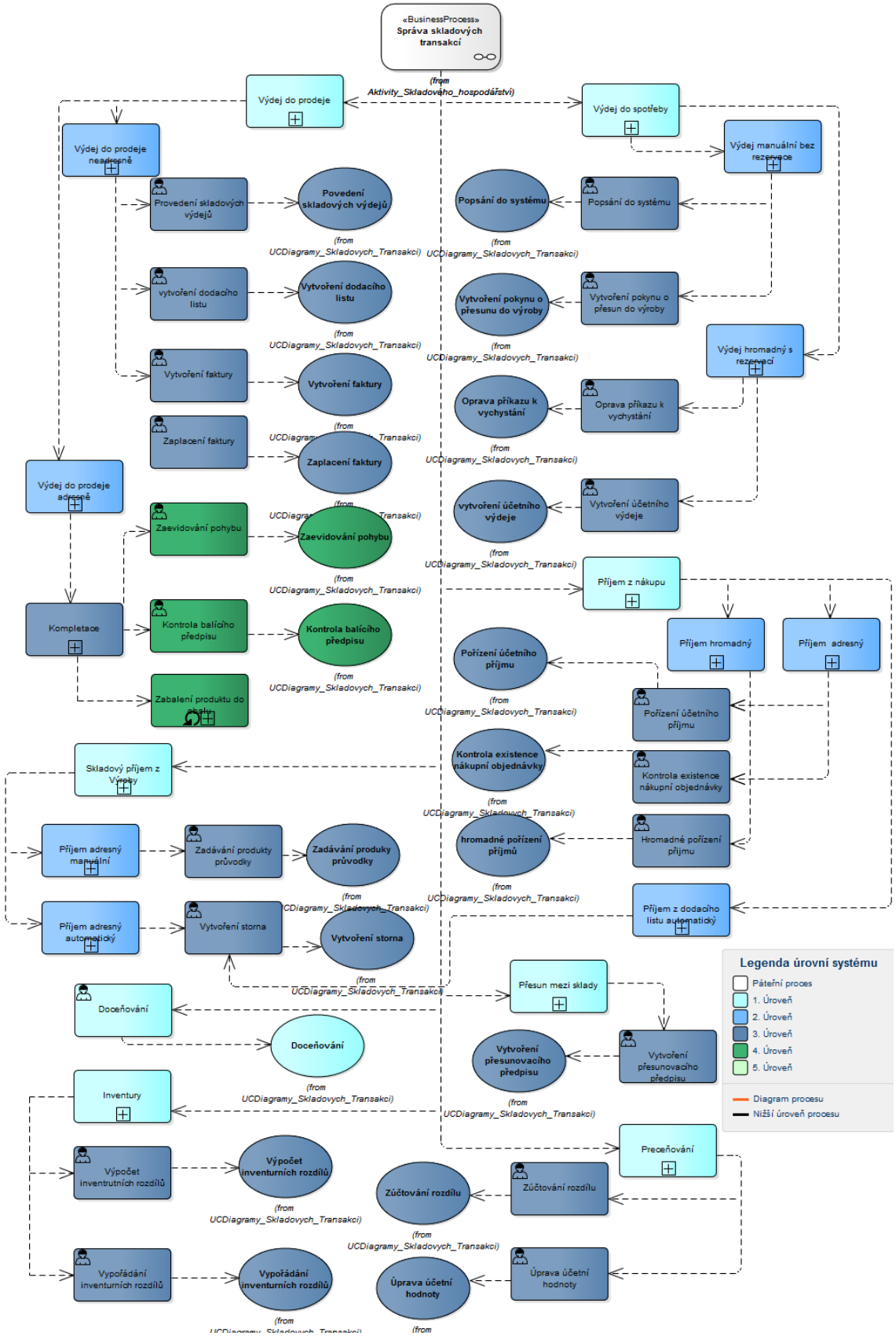
4.6 Skladové hospodářství - Potencionální Use case

Na základě výše uvedených procesů ve všech **Business** procesech v oblasti Skladové hospodářství můžeme jasně identifikovat typové úlohy. Tzn., že můžeme jasně definovat funkce systému zapojeného do skladového hospodářství. Jednou typovou úlohou může být požadovaná funkce, kterou chtějí naši zákazníci.

4.6.1 Identifikace potenciálních typových úloh ve správě skladových transakcí

Při zkoumání procesů si můžeme všimnout, že dané **Tasky** mají přiřazený i druh **Tasku**. Tyto přesné značky nám pomohou pochopit samotný systém. Všechny identifikační značky jsou popsány v kapitole 3.2. Při identifikaci typových úloh nás přesně zajímají tyto identifikační značky při rozlišování **Tasku**. V systému tak hledáme vždy identifikační značku typu **User**. Je vyobrazen jako panáček a značí, že v této aktivitě uživatel vytváří daný úkol v systému. Tato aktivita znamená, že v procesu pracuje člověk, který bude obsluhovat systém pro některý účel. Právě tato činnost musí být zmapovaná pro práci uživatele se systémem. Zmapované musí být kroky, podle kterých má uživatel v daném tasku postupovat, aby dosáhl cíle. Tyto kroky se píší do scénáře typových úloh. Pokud najdeme **Task**, který by odpovídal kritériím, v nichž kooperuje uživatel se systémem a tento „scénář“ kooperace je větší než jeden dva kroky, je nutné jej zmapovat do typové úlohy. Vše

je dobré si vysvětlit na příkladu. Vezměme si rovnou celou „Správu skladových transakcí“.



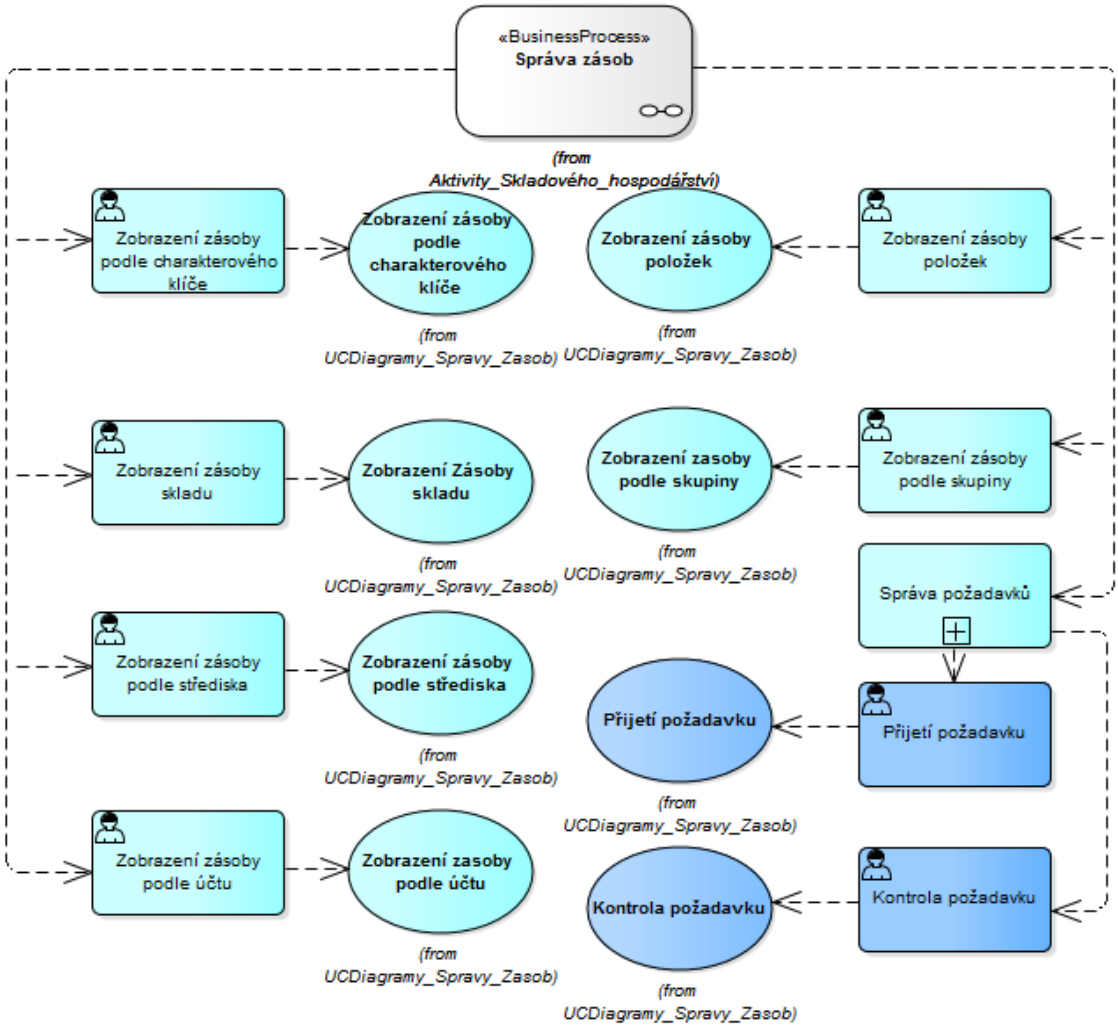
Obr. 77 Identifikace Typových úloh Zdroj: vlastní tvorba

Jak můžeme vidět z příkladu výše, ve stromu procesů se vytvořila mapa potencionálních **Tasků**, které byly typu **Task** a nemají primitivní scénář. Jelikož v této správě je mnoho diagramů, výsledný diagram tohoto mapování vypadá takto. Tento pohled byl zvolen z toho důvodu, že uživatel taktéž může vidět, na jaké úrovni dané typové úlohy je, a také v jakém procesu se nachází. Takto může čtenář jasně vidět, jaké typové úlohy se v rámci správy skladových transakcí vytvářejí. Má přehled nad všemi aktivitami, ve kterých se může vytvořit špatný doklad v rámci práce uživatele se systémem. Taktéž zde leží potenciální místo, kde se mohou optimalizovat skladové mechanismy. Pro nezasvěcené uživatele sice může tento stav vzbuzovat dojem, že tento systém je malý, avšak je nutné znovu zmínit, že se modelovaly jedny verze procesů, které jsou v systému. Pokud bychom modelovali všechny možné varianty, a ty dali do jednoho velkého diagramu tak, jak je to na obrázku č. 78., dostali bychom ohromný diagram čítající i několik desítek typových úloh a nespočet vnitřních aktivit a úrovní. Z obrázku výše tedy můžeme vypozorovat, že většina typových úloh je na druhé úrovni mapování, kromě typové úlohy doceňování, která je volána jako **Task** hned na první úrovni. Zato dvě typové úlohy „Kontrola balícího předpisu“ a „Zaevidování pohybu“ se vytvářejí až na 3 úrovni. Když modelujme zpětně systém bez rozsáhlých znalostí dané modelované oblasti, je dobré v tomto stavu konzultovat vzniklé typové úlohy s odborníkem z firmy, který dané oblasti rozumí. Nebo se podívat přímo do systému, zdali je tato funkce k dispozici. Pokud vše souhlasí, tak se přistoupí k dalšímu kroku vytváření typových úloh, a tím je vytváření scénářů.

4.6.2 Identifikace potenciálních typových úloh ve Správě zásob

Dále můžeme sledovat „Správu zásob“. Když nebereme v potaz procesy linkované ze skladových transakcí, tak můžeme definovat **use case** takto. Na obrázku níže je možné vidět, že jsme strom Správy zásob očistili od linkovaných procesů a ponechali pouze aktivity lokální. K těmto aktivitám typu **User** byl vytvořen **use case**. Tento **use case** bude následně popsán pomocí scénářů a bude vygenerována grafická podoba těchto scénářů, která může velice dobře sloužit pro simulování této funkce pro uživatele.

Situaci v úrovni typových úloh odlišnou od správy skladových transakcí vidíme na správě zásob. Zde jasně vidíme, že šest **use case** je znázorněno na první úrovni, kdežto pouze dvě typové úlohy na druhé úrovni. Když bychom opět rozpracovali celý **sub-proces** Správa požadavků s nalinkovanými typovými úlohami, dostali bychom ohromný diagram. Na to, jak jsou dílčí **sub-procesy** namodelovány, se můžeme podívat do Správy skladových transakcí. I když zde by ovšem opět byly na nižší úrovni, než je tomu ve správě skladových transakcí.

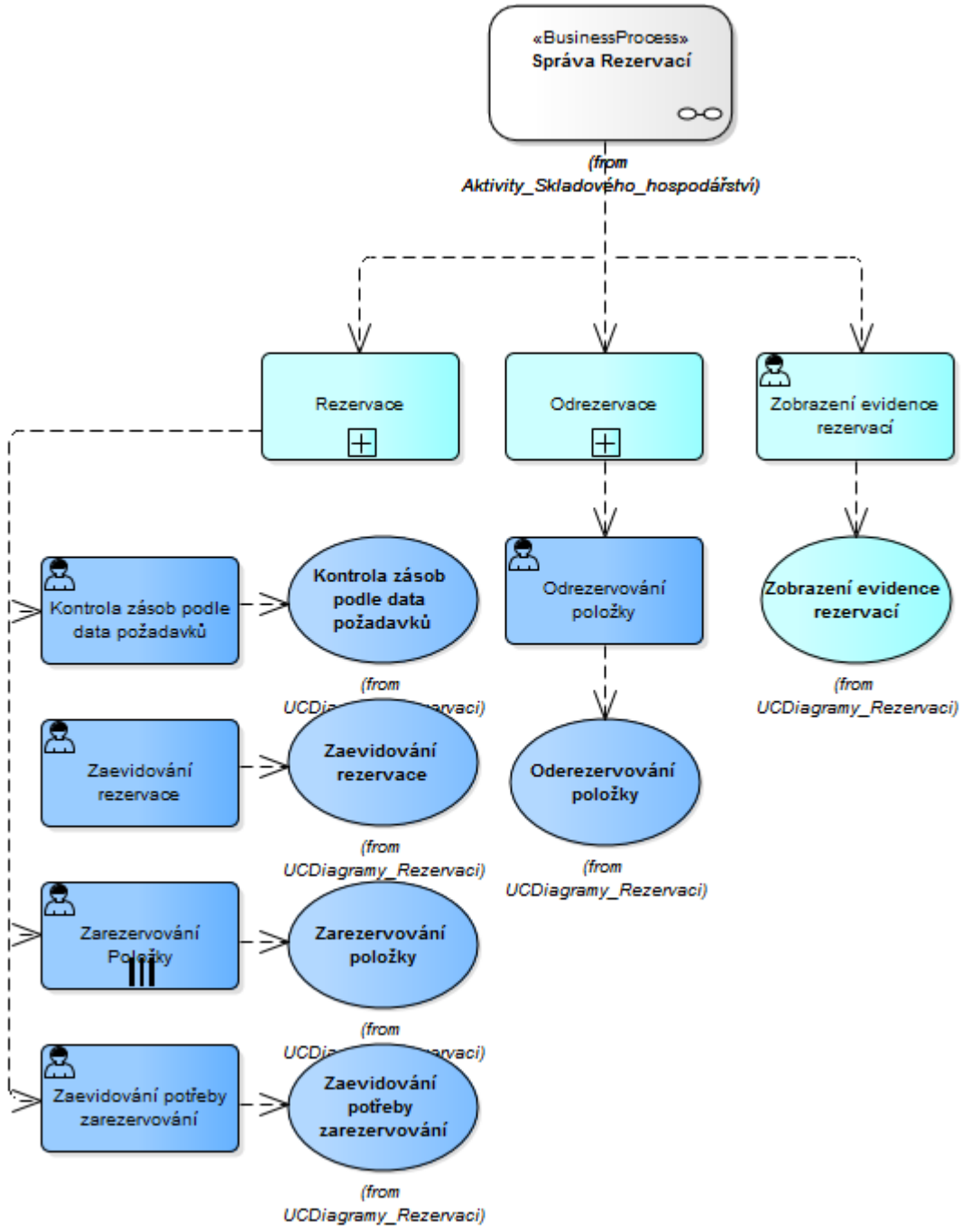


Obr. 78 Identifikace typových úloh ve správě zásob Zdroj: vlastní tvorba

4.6.3 Identifikace potenciálních typových úloh ve Správě rezervací

Podle tohoto principu poté vytvoříme pohled i na Správu rezervací, která má sice pouze 5 typových úloh, ale je nedílnou součástí skladového hospodářství. Zde se kontrolují stavy zásob, evidují rezervace a rezervují produkty pro určité

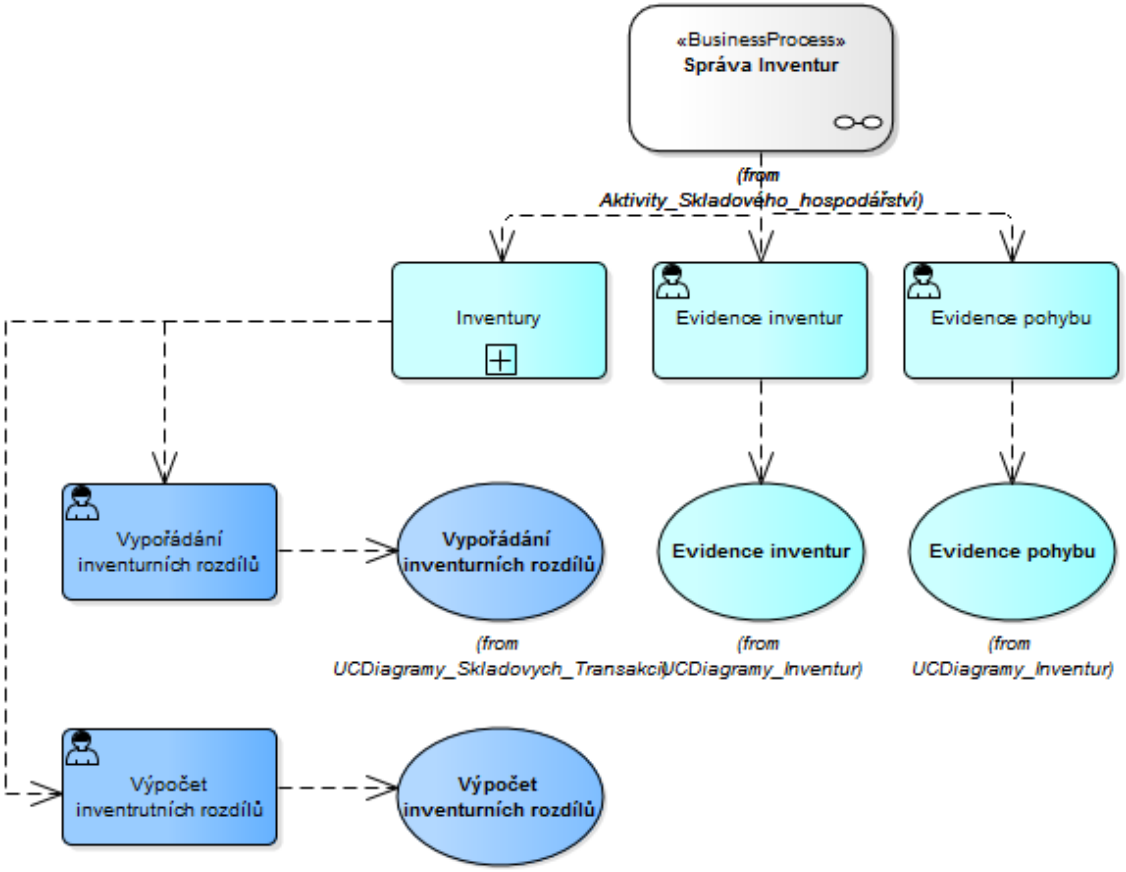
zákazníky. Také se zde evidují potřeby rezervací a odrezervují se zde produkty kvůli změně objednávky.



Obr. 79 Identifikace typových úloh v rezervací Zdroj: vlastní tvorba

4.6.4 Identifikace potenciálních typových úloh ve Správě inventur

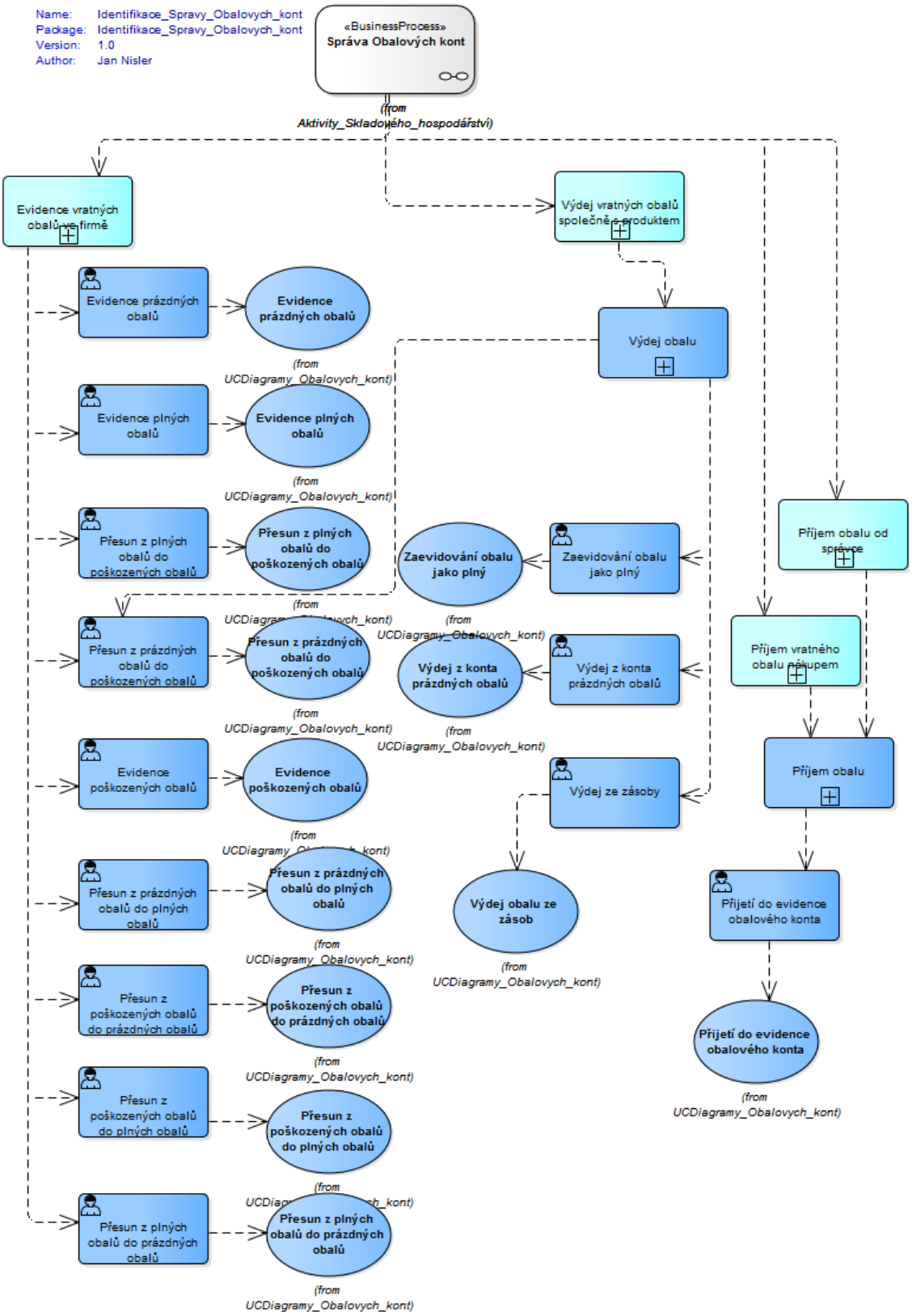
V této správě se evidují inventury a pohyby, které se během inventur vytvářely v rámci vypořádání inventurních rozdílů. Taktéž se podle pravidel porovnávají stavy zásob, které jsou v systému a fyzicky uskladněny na skladě.



Obr. 80 Identifikace typových úloh v inventurách Zdroj: vlastní tvorba

4.6.5 Identifikace potenciálních typových úloh ve Správě obalových kont

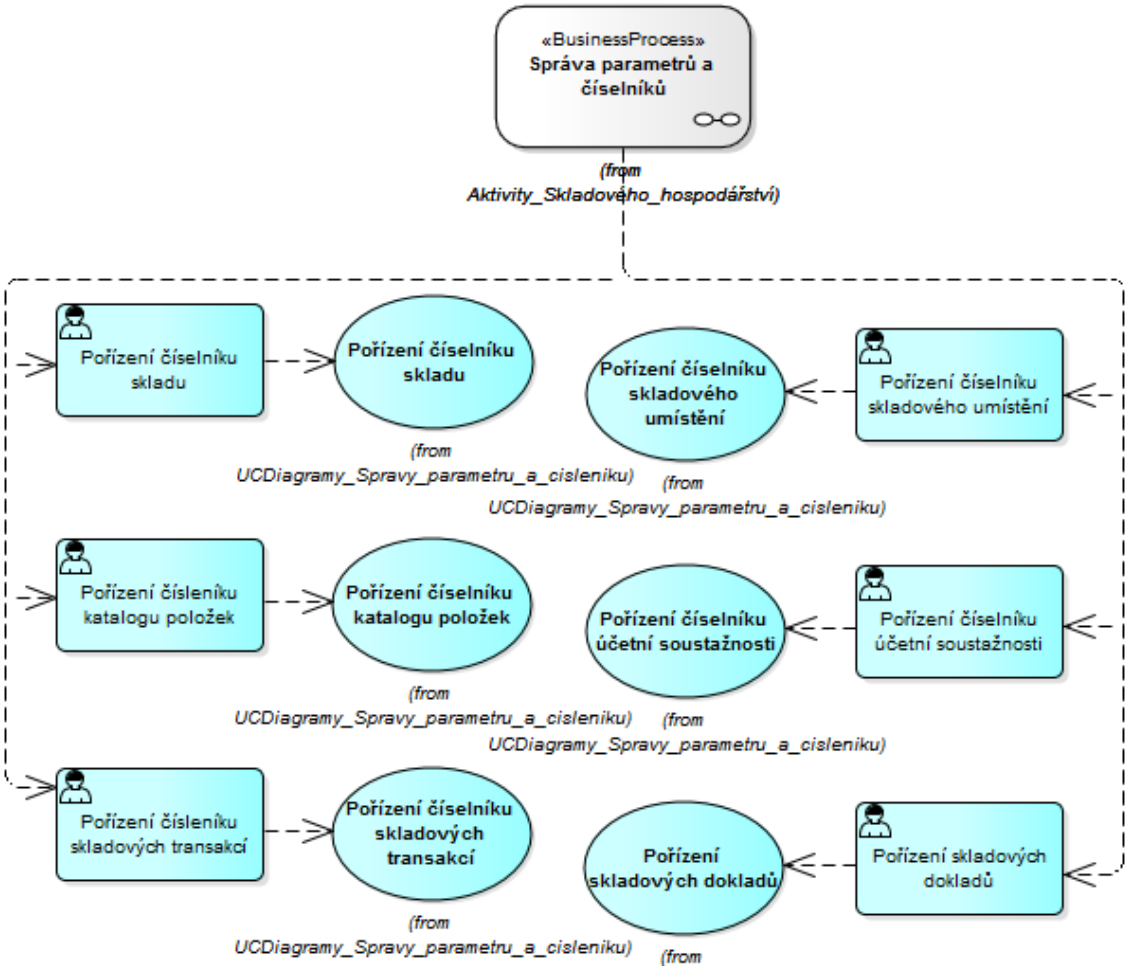
Ve Správě obalových kont může člověk vidět, kdy je ve dvou procesech stejný user **Task**. V procesu Výdej obalu a Evidence vratných obalů ve firmě se nachází user **Task** Přesun z prázdných obalů do poškozených obalů. Tento **Task** je proto napojen jednoduchou asociací na oba procesy, ve kterých se nachází. Jak je ale patrné z Aktivit rázu, Přesun z plných obalů do poškozených obalů nebo Přesun prázdných obalů do plných obalů, jsou tyto **User Tasky** velmi prosté, takže pro vyhotovení jejich **use case** by jeden scénář ani nestačil. Proto se v dalším kroku všechny tyto **Tasky** tohoto typu spojí v jeden **use case**, ve kterém bude charakterizován postup a výběr přesunu.



Obr. 81 Identifikace typových úloh ve správě obalových kont Zdroj: vlastní tvorba

4.6.6 Identifikace potenciálních typových úloh ve správě parametrů a číselníků

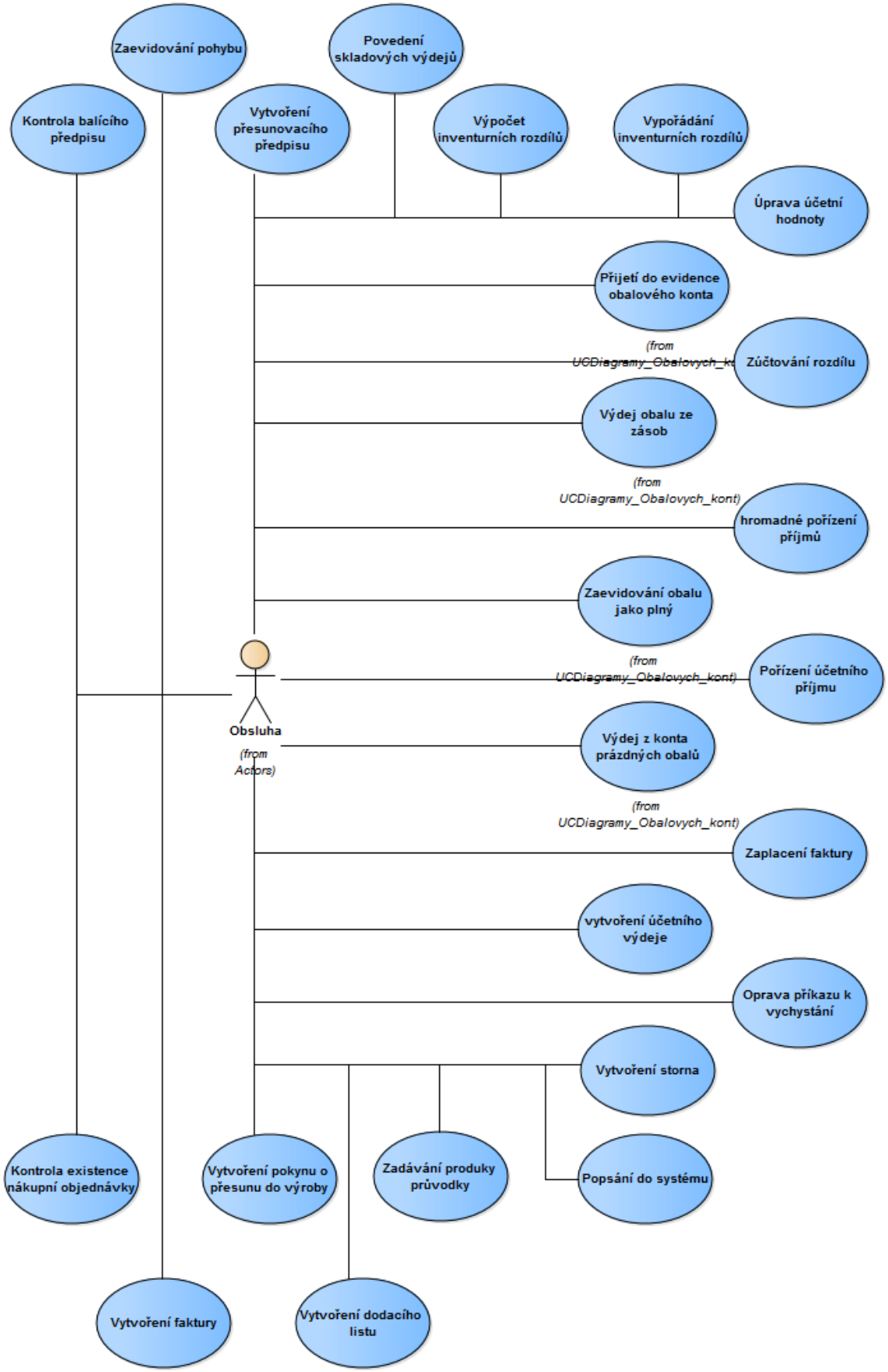
Jako poslední je business proces Správa parametrů a číselníků. Zde jsou typové úlohy pouze o nastavení parametrů a číselníků. Znovu mohou být tyto aktivity sjednoceny do jedné typové úlohy, která by plně sloužila jako přehled jednoho druhu scénáře, kde by bylo specifikováno o jaký číselník či parametr se jedná. V principu se jedná o ten samý druh scénáře, ovšem vyplňují se jiné údaje.



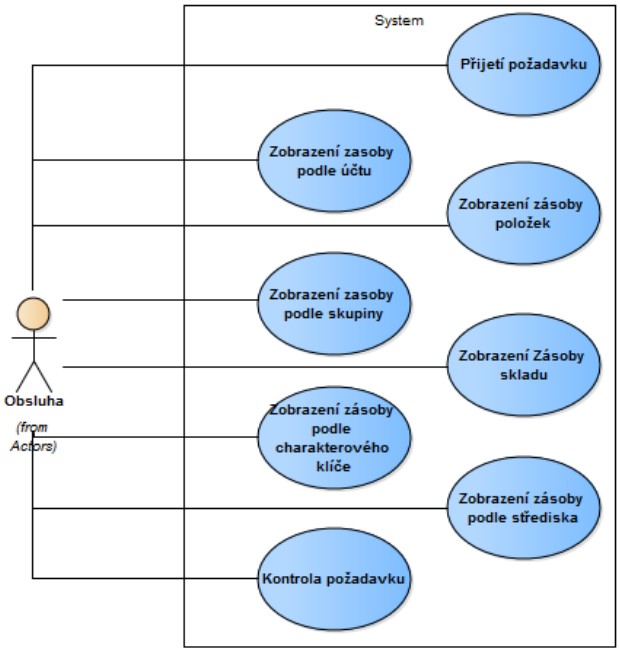
Obr. 82 Identifikace typových úloh ve Správě parametrů a číselníků Zdroj: vlastní tvorba

4.7 Skladové hospodářství - Přímé Use case

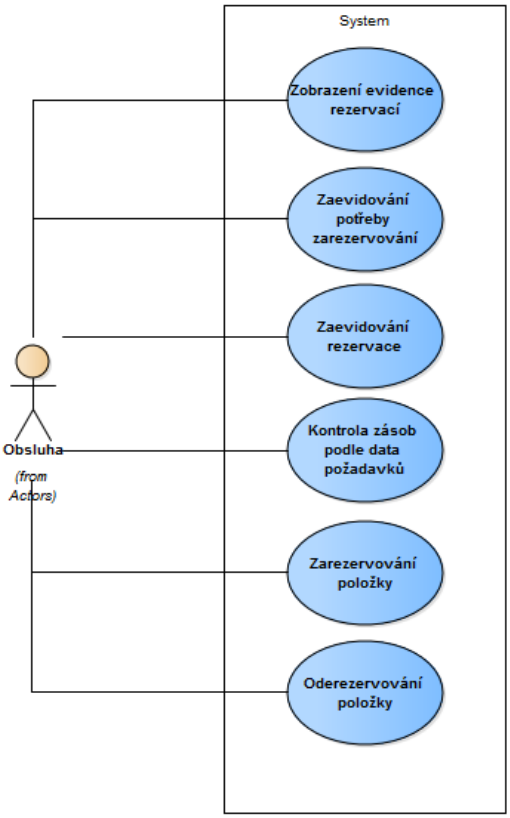
Nyní si uspořádáme **use case** do standardního diagramu, kde spojíme jednotlivé **use case** s aktéry, kteří je používají pro kooperaci se systémem.



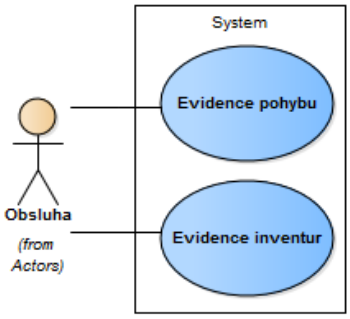
Obr. 83 Use case model pro Správu skladových transakcí Zdroj: vlastní tvorba



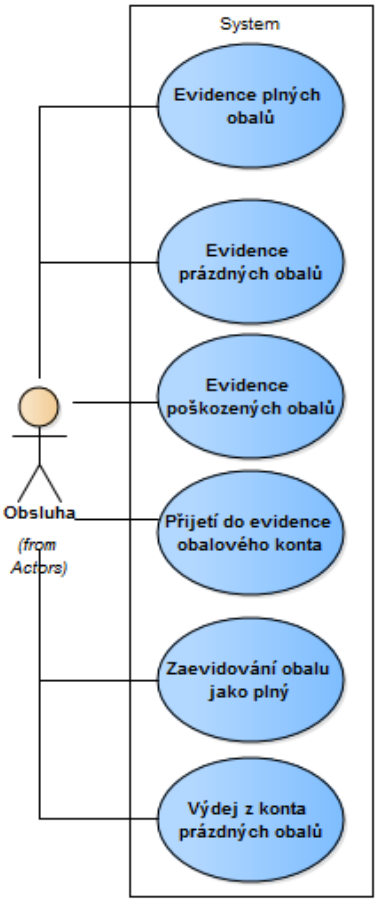
Obr. 85 Use case model pro Správu zásob Zdroj: vlastní tvorba

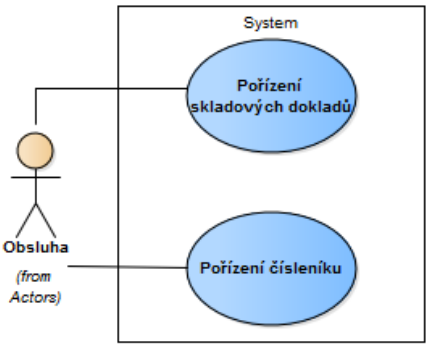


Obr. 84 Use case model pro Rezervací Zdroj: vlastní tvorba



Obr. 86 Strom Inventur Zdroj: vlastní tvorba





Obr. 88 Strom parametrů a číselníků Zdroj: vlastní tvorba

Obr. 87 Strom Obalových kont Zdroj: vlastní tvorba

Podle formátu, kdy jsme spojili skutečné **use case** s aktérem se vytvořily všechny správy **Business Processů** a jsou vyobrazeny na obrázcích č. 84. až po č. 88. Zde je možné vidět, že jsou všude použiti aktéři s názvem „Obsluha“. Tento univerzální název byl zvolen proto, že každá firma v praxi má jinou organizační strukturu ve skladech, tudíž si jej může každý doplnit podle vlastního předpisu. Ve většině případů se stává, že tyto **use case** jsou používány vícero uživateli. Jiný případ nastane, když jeden uživatel používá jen některé **use case** a druhý uživatel zase jiné. To záleží na formátu každé firmy, jak si tyto aktéry vytvoří. Když bychom poté vytvářeli scénáře, tak je můžeme vytvořit pomocí následujícího příkladu. Vezmeme jako příklad Správu zásob. Zde si ukážeme, jak se vytváří scénář a jak se vytváří **Alternate** či **Exeption path**. Následně si každý tento scénář vygenerujeme do grafické podoby a spustíme simulaci. Jako konečný krok bude vytvořena šablona pro generování do grafické podoby, kterou si my sami vytvoříme. V příloze poté bude část dokumentace obsažena.

4.8 Scénáře

Z důvodu velikosti následujících kroků si vybere pouze jeden **Business Process**, na kterém si ukážeme princip. Kdybychom takto popsali veškeré **use case** nacházející se na malém vzorku procesů, které se vytvořily, dostali bychom několika set stránkový dokument, což není naším záměrem. Jedním z hlavních cílů této práce bylo popsání skladového hospodářství pomocí procesů. Tyto dovětky jsou doplňující a rozvíjející samotné procesy. Vše poté může sloužit jako metodika pro ty, kteří si chtějí modelovat své procesy. Pro vytvoření tohoto kroku si vybereme Správu zásob. Tato správa čítá na 9 **use case**, na kterých se dobře ukáže princip scénářů, jejich změn a generování do grafické podoby.

V první řadě si vezmeme **use case** Zobrazení zásoby položek.

Tento jednoduchý **use case** je funkce, kdy se díváme na zásobu skladu do určité hloubky našeho zobrazení. Tento **use case** a ostatní, které se dívají na položky zásoby podle kritérií, nemohou být sjednoceny do stejného **use case**, jelikož vždy přistupujeme ke každé funkci jiným volání a jejich scénář je jiný. Podle pravidla „*Když se mění **use case** alespoň trochu od původního, musí vzniknout druhý*“. Proto i zde se dané **use case** rozdělily do několika.

4.8.1 Zobrazení Zásoby skladu

Vytvářené scénáře se píší ke každému use case do **Properties** >>> **Scenarios** >>> **Structured Scenario**. Zde se vytváří strukturovaný scénář, tedy každý krok se píše do jednotlivých kolonek. Zde se určuje číslo kroku, uživatele, stav a další. Pro nynější potřeby bude stačit pouze vytvořit kroky scénáře.

Popis typové úlohy:

Tento **use case** je použit pro zobrazení stavu zásob určitých položek na skladě. V rámci tohoto zobrazení se uživatel může dívat na dané položky podle další kritérií.

Scénář:

Basic path:

1. Obsluha v hlavním menu systému zvolí úlohu Zobrazení zásoby skladu
2. Zobrazí se seznam všech skladů, na kterých je evidována zásoba
3. Obsluha nastaví kurzor na sklad a zvolí příslušnou funkci dle toho, jaké informace o zásobě skladu chce získat - viz alternativní scénáře
4. Obsluha ukončí úlohu
5. Nastaví aplikaci do stavu před spuštěním Use Case

Alternate

Z hlavního scénáře můžeme ovšem vypozorovat, že za krok 4 bychom mohli ještě dát jiné kroky a to takové, že se uživatel může podívat na další členění položek. Všechny tyto varianty si popíšeme nyní:

Alternate1: **Funkce Zásoba výrobních čísel na skladě**

1. Obsluha zvolí funkci Výrobní čísla položky na skladě
2. Zobrazí se v řádkovém formuláři seznam všech výrobních čísel položky

Alternate2: **Funkce Zásoba šarží na skladě**

1. Obsluha zvolí funkci Šarže na skladě
2. Zobrazí se v řádkovém formuláři seznam všech šarží na skladě

Alternate3: **Funkce Zásoba položek na skladě**

1. Obsluha zvolí funkci Položky skladu
2. Zobrazí se v řádkovém formuláři seznam zásob všech položek skladu

Alternate4: **Funkce Zásoba palet na skladě**

1. Obsluha zvolí funkci Palety na skladě
2. Zobrazí se v řádkovém formuláři seznam všech palet na skladě

Alternate5: **Funkce Zásoba Balících jednotek na skladě**

1. Obsluha zvolí funkci Balící jednotky na skladě
2. Zobrazí se v řádkovém formuláři seznam všech balících jednotek na skladě

Alternate6: Funkce Zásoba balíků na skladě

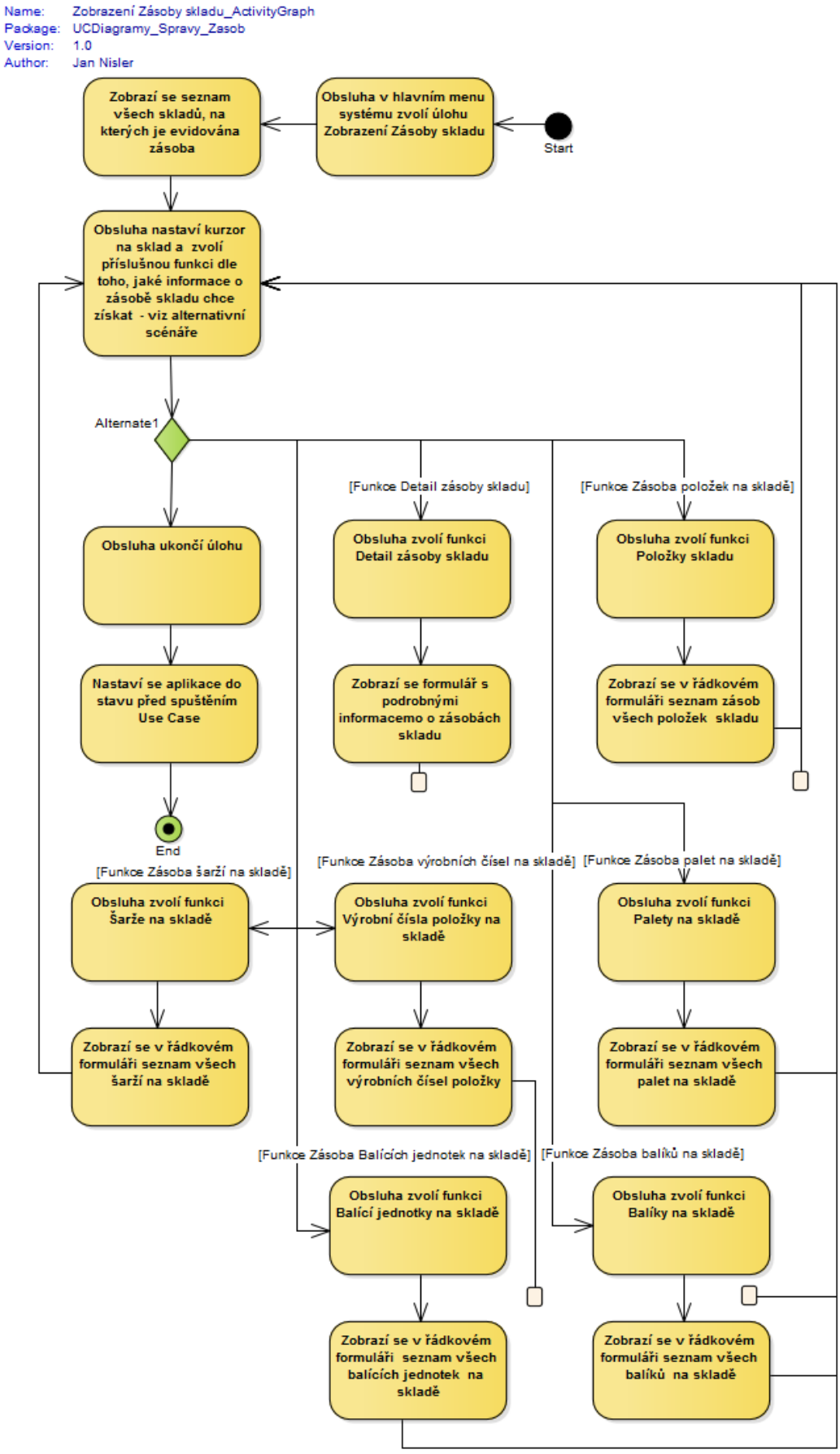
1. Obsluha zvolí funkci Balíky na skladě
2. Zobrazí se v řádkovém formuláři seznam všech balíků na skladě

Alternate7: **Funkce Detail zásoby skladu**

1. Obsluha zvolí funkci Detail zásoby skladu
2. Zobrazí se formulář s podrobnými informacemi o zásobách skladu

Tyto **Alternate** tedy budou napojeny na 4 krok hlavního scénáře, kde mohou začít místo původního kroku č. 5. Návrat těchto **Alternate** je stejný, jako jejich začátek, tedy za 4. krokem, jelikož, se může uživatel podívat třeba i na jiný pohled zásoby. Když máme takto vytvořený scénář, je dobré si jej představit i graficky. Mnohdy ze suchého textu nejsou zřetelné všechny odpovědi, které bychom chtěli. Toto generování do grafické podoby se vytváří v rámci jednoho **Use case** >>> **Properties** >>> **Scenario** >>> **Structured Scenario** >>> **Generate diagram**. Zde máme několik možností, jak můžeme daný scénář vygenerovat, třeba jako **State** diagram nebo jako aktivity diagram. Nyní je použit **Activity** diagram, který je pod syntaxí **UML**, která je popsána v kapitole v teoretické části této práce.

Na výsledném diagramu můžeme vidět jasné kroky v návaznosti na sebe a vstupy **Alternate** scénářů do hlavního scénáře. Takto se mohou vygenerovat všechny scénáře a poté se může provést i simulace těchto kroků. Simulace těchto kroků může být dobrým vysvětlujícím prostředkem pro pochopení jednotlivých kroků celého scénáře.



Obr. 89 Zobrazení zásoby skaldu grafický scénář Zdroj: vlastní tvorba

Ostatní scénáře se vytvářejí podobným způsobem, vždy tedy máme základní scénáře, které na začátku popíšeme. V tomto základním scénáři uvažujeme, že je všechno v pořádku a k danému cíli se dostaneme. Jedna typová funkce má mít vždy jeden cíl, který chce uživateli přinést. V horním případě, se tedy jedná o pohled na danou zásobu, kterou můžeme sledovat z mnoha úhlů. Je vhodné si ukázat ještě jeden **use case**. Tento **use case** se nazývá Zobrazení zásoby položek. Bohužel **Alternate** scénáře a **Exeption** scénáře se vytvářejí pouze na hlavní úrovni scénáře, tedy pouze na **Basic Path**. Vnoření **Alternate** do **Alternate** nelze, ani žádná kombinace nižších úrovní scénářů.

4.8.2 Zobrazení zásoby položek

Scénář se bude vytvářet do stejného místa jako předchozí příklad.

Popis typové úlohy: Tato typová úloha slouží pro pohled na položky naší zásoby, které jsou k dispozici v našem systému. Při pohledu na dané položky uvidíme veškeré souvislosti a doplňující informace, které daná položka vykazuje.

Scénář:

Basic path:

1. Obsluha zvolí úlohu pro zobrazení položky
2. Zobrazí se seznam všech druhů položek
3. Obsluha vybere jednu položku ze seznamu a potvrdí ji
4. Zobrazí se seznam položek vybraného druhu položky (umístění položky, šarže položky, skladová jednotka…)
5. Obsluha zvolí ukončení zobrazení
6. Uzavírá se formulář

Alternate

Z hlavního scénáře můžeme ovšem vypozorovat, že za krok 5 bychom mohli ještě vybrat jiné kroky a to takové, kdy se uživatel může podívat na další členění položek. Všechny tyto varianty si popíšeme nyní:

Alternate1: **Funkce Zásoba šarží na skladě**

1. Obsluha vybere funkci Šarže položky
2. Zobrazí se seznam všech šarží položky

Alternate2: **Funkce Paleta**

1. Obsluha vybere funkci Paleta Položky
2. Zobrazí se seznam všech palet příslušné položky

Alternate3: Funkce Balící jednotka

1. Obsluha vybere funkci Balící jednotka položky
2. Zobrazí se seznam balící jednotky příslušné položky

Alternate4: Funkce Balík

1. Obsluha vybere funkci Balík položky
2. Zobrazí se seznam balíků vybrané položky

Alternate5: Funkce výrobní čísla

1. Obsluha vybere funkci Výrobní číslo
2. Zobrazí se seznam výrobních čísel určité položky

Alternate6: Funkce Detail zásoby skladu

1. Obsluha vybere funkci Detail Položky
2. Zobrazí se karta s Detaily o položce (Datum výroby, Objednavatel, Dodavatel…)

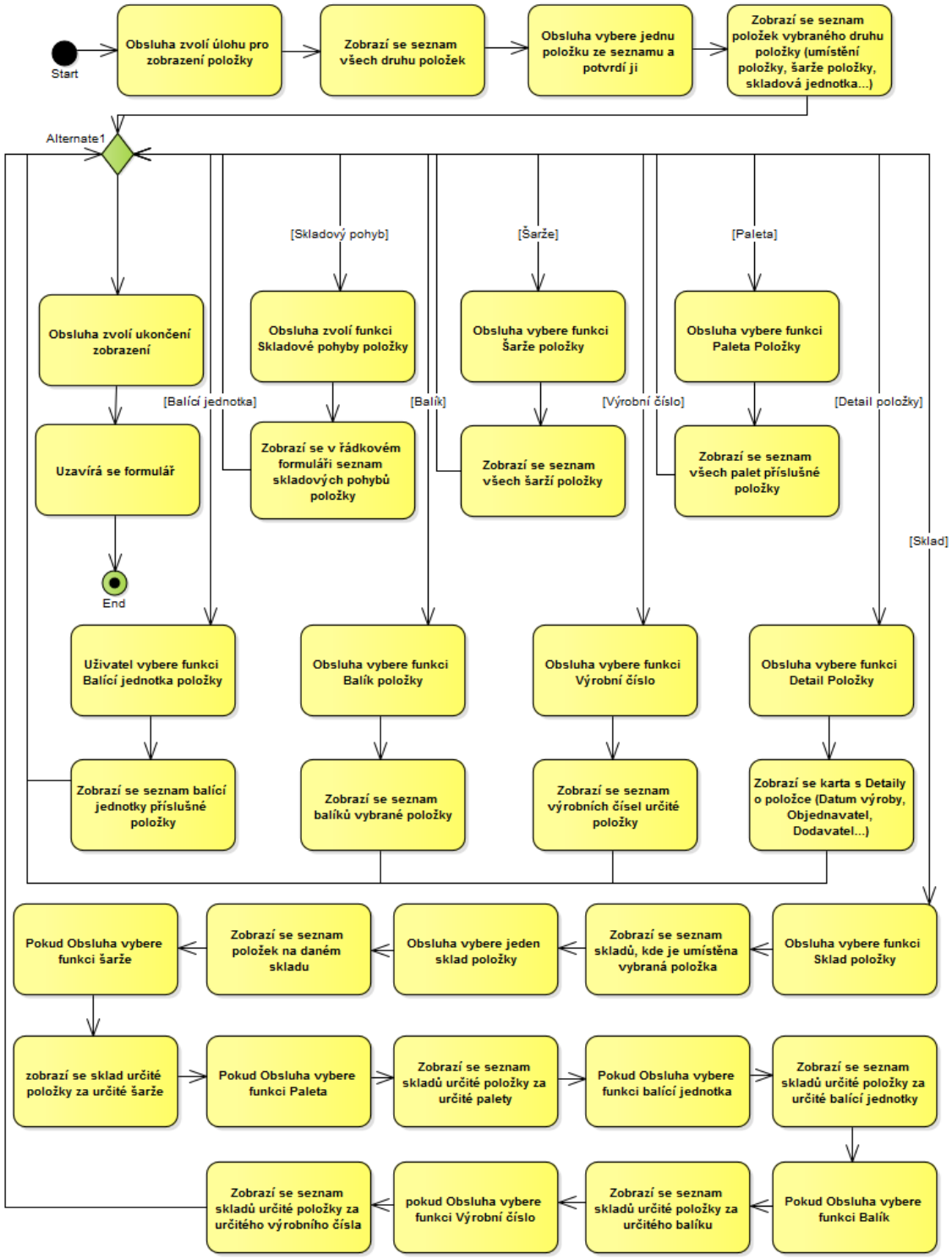
Alternate3: Funkce Skladový pohyb

1. Obsluha zvolí funkci Skladové pohyby položky
2. Zobrazí se v řádkovém formuláři seznam skladových pohybů položky

Alternate8: Sklad

1. Obsluha vybere funkci Sklad položky
2. Zobrazí se seznam skladů, kde je umístěna vybraná položka
3. Obsluha vybere jeden sklad položky
4. Zobrazí se seznam položek na daném skladu
5. Pokud Obsluha vybere funkci šarže
6. Zobrazí se sklad určité položky za určité šarže
7. Pokud Obsluha vybere funkci Paleta
8. Zobrazí se seznam skladů určité položky za určité palety
9. Pokud Obsluha vybere funkci balící jednotka
10. Zobrazí se seznam skladů určité položky za určité balící jednotky
11. Pokud Obsluha vybere funkci Balík
12. Zobrazí se seznam skladů určité položky za určitého balíku
13. pokud Obsluha vybere funkci Výrobní číslo

14. Zobrazí se seznam skladů určité položky za určitého výrobního čísla

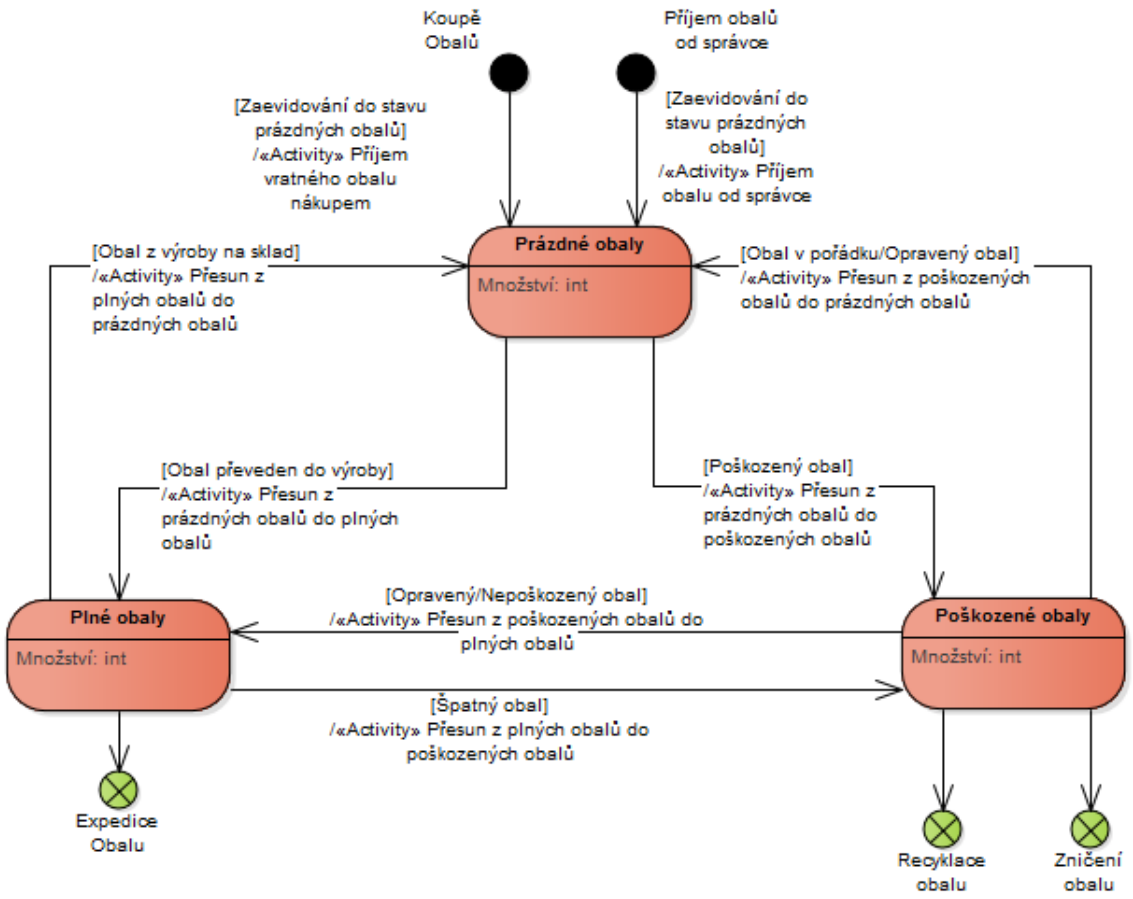


Obr. 90 Zobrazení zásoby položek Zdroj: vlastní tvorba

Opět tyto alternativní scénáře mohou být vyvolány místo bodu 5 a vracejí se zpět na rozhodovací uzel, kde se znovu uživatel rozhoduje jakou funkci zvolit, nebo

ukončit celý formulář. Právě v alternativním scénáři číslo 8, můžeme vidět problém, který byl předložen na začátku tohoto příkladu. Nástroj nám nedovolí vnořit **Alternate** scénář či **Exception** scénář do již vnořeného scénářů obou typů. Z tohoto důvodu, když chceme ukázat na to, že se v daném use case můžeme vnořovat do několika úrovní, musíme vždy skončit na úrovni **Alternate**, tedy druhé a zbylé podmínky napsat pomocí principu „pokud“, viz **Alternate** scénář číslo 8.

Pomocí těchto principů si každou funkci v systému můžeme vymodelovat a mít ji jako podklad pro další vývoj svého systému. K současnému stavu taktéž můžeme přiřadit ještě další pohledy, jak se dívat na daný systém, například stavový diagram. Tento diagram může dále rozvíjet povědomí o daných objektech. Například máme Obalová konta. V těchto kontech jsou vratné obaly a obaly na jedno použití. Jaké u nich vlastně máme stavy? Jak se mezi jednotlivými stavy pohybuje obal, jak do těchto přichází a jak odchází? Známe tyto principy a kroky?



Obr. 91 Stavový diagram Zdroj: vlastní tvorba

Zde můžeme vidět Stavový diagram a jednotlivé přechody mezi stavy. Všimněme si taktéž, že v přechodech jsou aktivity, ve kterých se mění stav daného objektu na nový stav. Tyto aktivity už jsme si modelovali na začátku příkladu o skladovém hospodářství. Všechny tyto modely jsou propojené a jejich tvorba není nikterak těžká. Nejdůležitějším faktorem je to, abychom znali dobře dané funkce a procesy, ve kterých se tyto vše modely vytvářejí.

4.9 Generování dokumentace

Když se vymodelovaly všechny potřebné modely do grafické podoby, je taktéž dobré si vytvořené modely prohlédnout. Nejlepší pohled na diagramy je samozřejmě přes prostředí **EA**. Ovšem když vytváříme tyto modely a potřebujeme je někomu prezentovat, ukázat či vytisknout, nemůžeme předpokládat, že má na svém počítači **EA**. Na světě je mnoho **Case** nástrojů a každý nemusí mít nainstalovaný kompatibilní nástroj. Proto je v nástroji **EA** vytvořené prostředí pro tvorbu dokumentace. Dokumentace může být ve formě textového výstupu v podobě pdf. docx či rtf. Taktéž můžeme model vygenerovat do podoby **HTML**. Zde si daný model můžeme promítat podobně, jako v nástroji **EA**. Prostředí pro generování Dokumentace je dobře zvládnuté a pro své generování používá právě šablonu, kde se nacházejí **Tagy**, podle kterých zkoumá projekt, a místo **Tagů** přímo doplňuje jednotlivé prvky modelu. Takto vzniká celá dokumentace.

4.9.1 Šablona pro generování dokumentace

Každý si může generovat dokumentaci nad každou úrovní **Package** a to pomocí předem nachystaných šablon, které jsou už v základu **EA**. Ovšem tyto šablony jsou v angličtině a pro zdejší podmínky jsou přesycené zbytečnými informacemi. Proto nejlepší šablonou je vždy ta, kterou si vytvoříme sami. Jak ji vytvořit se nyní dozvíme.

Otevření šablony

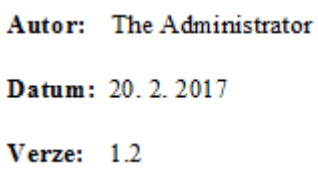
Nyní si musíme otevřít prostředí, kde si danou šablonu budeme vytvářet. Otevřeme si **View** >>> **Portals** >>> **Publish** >>> **Edit Templates** >>> **Document Templates**. Nyní se dostáváme do tvorby vnitřní části šablony. Buď můžeme začít vytvářet obsah šablony, nebo si zvolíme novou šablonu, kde si vybereme typ

šablony a název. Typ šablony se volí proto, že EA používá pro generování 3 šablony.

První šablona

První částí je úvodní strana šablony. V této části si nadefinujeme, jak má vypadat úvodní strana celé dokumentace. Do prostoru volného listu si pravým kliknutím >>> **Report Constants** můžeme volit informace, které bychom chtěli vždy vidět. Například si zvolíme **ReportAuthor**, **ReportDataShort** a **ReportVersion**. Z těchto informací získáme autora, datum vzniku dokumentace a verze projektu viz obrázek č. 97. Po vygenerování v projektu dostáváme třeba takové údaje, viz obrázek č. 98.

Obrázek 97 index zdroj: vlastní tvorba



Obrázek 98 výsledek indexu zdroj: vlastní tvorba

Samozřejmě potřebujeme mít na úvodní straně název projektu, který jsme generovali a například i program, ve kterém jsme modelovali. Když vytváříme dokumentaci pro zákazníka, nesmíme zapomenout i grafický znak jeho firmy nebo znaky naší a jejich firmy. Pro jednoduchost jsme zvolili obrázek fakulty. Pro lepší orientaci na úvodní straně je vhodné vytvořit si tabulku bez ohraničení a v ní zadávat jednotlivé údaje. Takto můžeme mnohem pohodlněji pracovat na úvodní straně. Vše si můžeme prohlédnout v příloze č. 3. kde je vytvořená kompletně úvodní šablona pro generování úvodní strany dokumentace.

Druhá šablona

Druhou šablonou, kterou si vytvoříme, je Osnova. Tato šablona definuje vzhled a uzpůsobení osnovy dané generované dokumentace. (V této šabloně je dobré vycházet z již vytvořené osnovy, kde upravíme název do české podoby a jinak není na osnově co řešit).

Třetí šablona

Třetí a nejdůležitější částí je vytvoření těla šablony. V této části definujeme, jaké informace budeme chtít generovat. Otevřeme si tedy prostředí pro vytvoření šablony pro tvorbu. Nyní si musíme definovat, jak by daná dokumentace měla

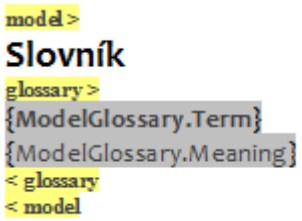
vypadat. Pro názorný příklad nyní budeme vycházet z našich potřeb, které si nyní definujeme.

Požadavky na dokument

- Dokument bude definován jasným názvem, který bude zřejmý ze všech stran dokumentu
- Šablonu, ze které jsme dokument generovali
- Cestu dokumentu, kde mám daný dokument uložen v počítači
- Nadefinovaný slovník
- Odsazení **Package** od druhého **Package**
- **Notes Package**
- Obrázek Diagramu a jeho **Notes**
- Elementy Diagramu a jejich popisky a druh elementu
- Scénáře u **use case**
- Další úrovně mají mít stejné vlastnosti jako kořenové **Package**

Tvorba šablona- Slovník

Podle těchto parametrů si nyní vybereme parametry z levého sloupce. Začneme od definování slovníku. Tento slovník nám udává sjednocenou terminologii, kterou jsme si definovali se zákazníkem či partnerem. Proto musí být na začátku celého dokumentu. Zaškrtneme si proto na prvním místě **Model** a **Glossary**. Nyní se nám objevili v dokumentu texty v žlutém poli. To jsou právě zmíněné **Tagy**. Nyní je dobré si daný slovník nadepsat a definovat, které **Tagy** chceme mít ve slovníku. Název napíšeme mezi **Tagy Model>** a **Glossary>** a terminologii slovníku zase mezi **glossary>** a **<glossary>**.



Obrázek 99. slovník zdroj: vlastní tvorba

Výsledek vypadá tak, jak je to uvedeno na obrázku č. 99. Vždy, když vytváříme nadpis daného elementu, musíme si uvědomit, pro jaký účel jej vytváříme a na jakou úroveň chceme daný pojem dát. Například když si vezmeme nadpis „Slovník“. Je vytvořen před daný **Tag** z toho důvodu, že jej chceme vidět vždy jednou a to nad všemi termíny, které jsou použity

ve slovníku. Když bychom dali nadpis až mezi **Tagy**, vznikl by nám z toho slovník, který by měl před každým termínem nadpis Slovník, což je nežádoucí. **Tagy** se berou uzavřeně, a tak když zkoumají projekt ohledně daných pojmů, vezmou vždy tělo **Tagu** a do něj vloží nalezený termín. Tento fakt je dobré mít na paměti, jelikož při tvorbě dalších částí těla šablony se nám tyto poznatky budou hodit. Je vhodné oddělit slovník od dalších částí, takže za zmíněné **Tagy** vložím **Page Break**, což znamená, že další část začne vždy na nové straně. Stačí na místo, kde chceme lomit text (přímo na další řádek po **Tagu**) pravým klikem myši do řádku >>> **Insert** >>> **Insert Break** >>> **Page Break**. Takto se nám vždy oddělí slovník od další části. Automaticky se nám v šabloně vytvoří linie **Page break** a vytvoří se nám nová stránka.

Tvorba šablony - Vybrání tagů

Nyní si vybereme z pravého menu všechny **Tagy**, které bychom chtěl v dokumentaci mít.

Podle zadaných požadavků si tak vybereme **Tag Package**, který je hlavní **Tag.** Bez tohoto **Tagu** nemůže vytvářet dílčí **Tagy**. Hned poté vybereme **Tag** Package Element. U **Tagu** Diagram vybereme i jeho vnitřní **Tag** Element. Díky tomuto tagu se nám budou generovat elementy, které se nachází v diagramu, i když jsou nalinkované z jiného místa projektu. Element v **Project Browseru** je jiný element než element diagramu. Tento fakt musíme respektovat obzvlášť při generování dokumentace. Dalším **Tagem**, který vybereme, je opět Element, který je na úrovni **Tagu** Diagram. Zde vybereme vnitřní **Tag Scenario**, jeho vnitřní **Tagy Structured Scenarios** a jeho vnitřní **Exception**. Dalším vnitřním **Tagem Tagu** Elementu vybereme **Tag** Diagram a **Child Elements**. Vybereme ještě **Tag** na úrovni Element, a to **Child Package**. Díky tomuto **Tagu** se budou generovat i podúrovně kořenového **Package**. Nyní máme všechny **Tagy**, které byly požadované, vybrané.

Nyní si mezi vybrané **Tagy** vytvoříme ještě Inserty, které upřesní, jaké informace chceme generovat z projektu do dokumentu. Například hned první **Tag Package**. Na prvním místě musíme určit Název **Package**. To uděláme tak, že pravým kliknutím na požadované místo >>> **Insert Field** >>> **Name**. Nyní se nám vytvoří **Insert Pkg.Name**. To znamená, že se nám vygenerují jména procházejících

Packages, které máme v projektu. Podle požadavků ještě musíme určit i poznámky **Package**. To uděláme obdobně jako u jména, pouze nyní vybereme Notes.

Nyní přejdeme k dalšímu **tagu**. V diagramu, jak bylo uvedeno v požadavcích, potřebujeme vidět obrázek diagramu a název diagramu. Takže si obdobným způsobem jako v předchozích krocích vytvoříme jméno diagramu. Obrázek diagramu je vhodné vložit doprostřed stránky, proto u tohoto Insertu vybereme středové řádkování. Při vícero obrázcích je vhodné číslovat obrázky. Takže si pod **tag Diagram.DiagramImg** vložíme ještě **tag DiagramFigure**.

Jako další **Tag,** který si uděláme, je Element. Tedy přesněji řečeno oba Tagy Element. Někomu to může připadat matoucí, proč jsou tam **Tagy** dva a ještě více matoucí, protože budou mít skoro stejný popis. Proč tomu tak je? Musíme si uvědomit, že první **Tag** element je element diagramu, takže se nám vygenerují elementy diagramu. Druhý **Tag** je pro elementy **Package**. Jsou to tedy dva rozdílné **Tagy**, ovšem plní stejný účel, vygenerují se nám elementy z požadovaného místa. Nyní možná může nastat otázka „Ale, když máme dva **Tagy** stejného rázu, nevygenerují se nám dvakrát tytéž elementy?“ Odpověď je nikoliv. Při generování v tomto úseku dojde k porovnání elementů **Package** a diagramu a pokud jsou v diagramu stejné elementy jako v **Package**, vygenerují se vždy jako elementy **Package**, a k elementu diagramu se nevytvoří. Ovšem když je element nalinkovaný, vytvoří se tento element k diagramu. Takže zde má přednost element **Package** a generuje se do tohoto místa. Tohle řešení je velmi nevhodné a bylo by dobré jej nahradit tak, aby se upřednostňoval diagram nad **Pakcage**. Toto řešení by ale bylo vhodné pouze zde. Vývojáři měli určitě důvod, proč tento způsob zvolili. Nyní přejdeme k **Tagu** Element. Zde je vhodné určit druh elementu jeho název a poznámky. Proto je vhodné si vytvořit neviditelnou tabulku. Pravým kliknutím na místo kde chceme tabulku >>> Table >>> Insert table. Volíme tak, abychom měli 1 **Rows** (řádek) a 2 **Columns** (sloupce, buňky). Neviditelná tabulka jinak znamená, že nemá žádné ohraničení. Toto pravidlo splňuje vytvořená tabulka už v základu, takže se nemusí nic upravovat. Do první buňky vložíme **Element.Type** a do druhé **Element.Name**. Takto máme odlišený typ a jméno elementu. Je vhodné si ještě upravit šířku buněk, abychom je neměli až v prostředku strany. Za tabulku vložíme notes elementu. Tyto stejné úpravy uděláme i pro druhý **Tag** element. Je vhodné

vytvořit inserty ručně než je kopírovat. Prostředí šablony není doladěné na tyto postupy a má tendenci se zasekávat a uzamknout **Tagy**. Nyní si vytvoříme pro případy use case ještě scénáře. To uděláme obdobně jako pro element. Jelikož máme několik druhů scénářů, musíme je dobře odlišit. Vytvoříme si znovu tabulku nyní se dvěma řádky a vrchní řádek rozdělíme na dva sloupce. Vložíme tedy opět **Elemscenario.Type** a **ElemScenario.Scenario**. Do prvního a druhého řádku, bychom si měli přetáhnout **Tagy**, či je znovu vložit **Structured scenario** a **Exception.** kde u **Tagu Structured scenario** si určíme Inserty Step, tedy krok scénáře a **Action** tedy akci kroku scénáře. Do **Exception** si vložíme opět **Type,Step.Name** a **Join**. Tyto **Exception** znamenají, že do scénáře může vstupovat i další alternativní scénář, který bude mít příchod do scénáře označen. Je dobré ještě před insertem **Join** zvolit větu „-návrat k bodu scénáře“. Protože **Join** znamená, že se po daném alternativním scénáři uživatel vrátí na určité místo scénáře. Toto místo může být konec hlavního scénáře nebo jakýkoliv bod scénáře, který ležel před bodem, kdy se scénář rozdělil. Takto máme vytvořené všechny **Tagy** a Inserty v dokumentaci.

Nyní je dobré si podle požadavků určit ještě na každé straně název dokumentu a šablonu. Toto je dobré vložit do záhlaví a zápatí strany. Jednoduše si jej můžeme vytvořit na jakékoliv straně těla šablon. Pravým tlačítkem do prostoru strany >>> **Edit** >>> **Edit page Header/ Footer**. Nyní máme možnost upravit záhlaví a zápatí. Zde je znovu dobré vytvořit si tabulku, nyní se 3 sloupci a jedním řádkem. Na záhlaví vlevo je vhodné vložit jméno pravým stiskem do buňky >>> **Report Constants** >>> **ReportName**. Do pravého rohu je dobré vložit datum vzniku dokumentace. Postupujeme obdobně jako u jména, jen místo jména vložíme **ReportDataShort**. V zápatí je dobré v prostřední buňce vytvořit číslování stran. Pravým stiskem na buňku >>> **Insert** >>> **Page Number**. Do pravého rohu je zde vhodné vložit šablonu, kterou jsme použili pro generování. Šablona se nazývá pod názvem **ReportTitle**. Nyní máme vše vytvořené.

4.9.2 Šablona pro generování dokumentace

Nyní si pouze vyberme požadovaný **Package z Project Browseru** pravým kliknutím na daný **Package** >>> **Documentation** >>> **Generate Documentation**.

Zde si zvolíme námi vybrané šablony. Je ještě dobré když máme nepojmenované elementy v záložce **Options** zvolit **Hide <Anonymous> elements**. Tato funkce zabrání generování elementu beze jména. Poté nám stačí vybrat adresář, kam vygenerujeme dokumentaci a potvrdíme generování **Generate**. Abychom viděli rovnou vytvořený dokument, je dobré si zaškrtnout **Check box Use Internal Viewer**. Nyní máme vše hotové a námi vytvořený projekt je vygenerovaný do námi zvolené podoby. Vytvořenou šablonu a část dokumentace je možné vidět v příloze č. 3.

5 Shrnutí výsledků

Při vypracování této práce bylo velmi obtížné především získávání informací, a to zejména z toho důvodu, že neexistuje jednotná publikace, která by obsáhla samotné skladové hospodářství. V mnoha verzích podobných článků v časopisech či publikacích, zabývajících se výrobou či logistikou, se mluví vždy o skladovém hospodářství z rozdílných úhlů pohledu. Navíc každý autor si vytváří svůj názor na danou oblast, proto volba té nejlepší varianty popisu je nelehkým úkolem. Stanoveného cíle této práce bylo dosaženo ve velké míře, zejména ve vyjádření jasné funkce a principu skladu. Další částí, která zde byla zmiňována, byla syntaxe **BPMN**. Tato syntaxe byla obtížnější z důvodu nejednoznačnosti tuzemských či zahraničních publikací. Z popsání historie syntaxe a vyjádření jednotlivých elementů se tak stal běh na dlouho trať, kdy pro získání hodnotných informací bylo nutné prostudovat několik desítek publikací a hledat v nich ty správné informace. To se ve výsledku podařilo a daná syntaxe je do podrobných informací popsána. Popsání nástroje **Enterprise Architect** zde nebylo prioritou, proto je jeho popis velmi stručný, pouze na úrovni základního popisu. Pro praktickou část bylo nutné ve firmě získat informace o dané oblasti. Tyto informace byly získávány a dolaďovány v rámci několik měsíců. I tak nelze říci, že daný výsledek je finální, jelikož se přestalo modelovat na abstraktní úrovni. Pro hloubkové modelování by bylo nutné získat a osvojit si informace zahrnující celou oblast v daném systému. Tyto modely tak musí tvořit pouze odborníci, aby byly obsaženy všechny možné modely. I při handicapu nedostatečné hloubky tak vznikl užitečný pomocník pro

firmy pro modelování. Další možný rozvoj už nyní leží pouze na nich. Na závěr praktické části byly ukázány další potencionální pohledy na systém, které by měly sloužit vývojářům jako podnět pro vytvoření vlastního názoru. Na základě vlastních zkušeností, které již v modelování autor získal, lze konstatovat, že tyto modely jsou nejlepším způsobem, jak udržet krok ve vývoji systému. Současně nám dané postupy dávají možnost veškeré dokumentace. Tento fakt je velice důležitý.

6 Závěry a doporučení

Každá firma si své vlastní **Know-how** střeží, jelikož jeho vytvoření stojí firmy nemalé peníze. Proto vznikl tento dokument, jako první nasazení a promýšlení podnikových procesů do grafické podoby v rámci své vlastní práce. I když tento dokument nemůže nikterak nahradit celou obsáhlou sofistikovanost celého pojetí podnikových procesů v rámci grafického mapování, poskytuje dostatečné informace pro firmy, na základě kterých mají možnost pochopit a optimalizovat všechny své procesy podle potřeby. Grafické mapování je nyní nejenom nutností pro další přežití firem, ale toto mapování už je v dnešním světě standardem. Nástrojů, které umí mapovat podnikové procesy, je mnoho. Zde byl použit **Enterprise Architect**, který je vřele doporučován pro svůj velký repertoár funkcí, které se při tvorbě a řízení projektů velice hodí. Dalším nástrojem promodelování je syntaxe. Zde byla popsána syntaxe **BPMN**, která je doporučována pro svůj široký depozitář elementů, které při správném uchopení dokáží popsat veškeré procesy, které mohou existovat. Bohužel právě k této syntaxi existuje mnoho teorií a materiálů, které víceméně popisují samotnou syntaxi, ale nedávají nám jednotnou metodiku, jak danou syntaxi uchopit a chápat ji do těch nejmenších detailů. To je v závislosti na otevřenosti syntaxe pochopitelné, ale pro kooperaci partnerů, kteří svou syntaxi používají podle sebe, je tento stav velmi nežádoucí a matoucí. Tento postup má v praktickém světě velký potenciál. Toto tvrzení lze doložit tím, že v současné době už dané postupy používají velké firmy pro svůj vývoj v informačním světě, jako je například Škoda Auto, IBM nebo Assecco atd. Závěry v této práci taktéž už uchopili v rámci našeho partnerství 3 firmy. Sjednocení na bázi modelování je k pro partnerství velice důležité. Tyto firmy používají

komplexní přístup, který by zde nebylo možné popsat, jelikož sám o sobě čítá bezmála 300 stran a stále nejsou postupy úplně dopracovány do kýženého stavu. Prostřednictvím této práce tak dává autor prostor pro vlastní náhled na tuto problematiku a sám navrhuje postup, jak vylepšit a mapovat podnikové procesy právě v rámci grafického mapování. Nic není jednoduché, ale přidaná hodnota je tak obrovského rozměru, že je dobré se tímto přístupem vést a zdokonalit své vývojové schopnosti a efektivitu. Další průběh vylepšování vlastních procesů může být i simulace podnikových procesů. Tyto simulace nám mohou ukázat, jak se mohou procesy optimalizovat tak, aby efektivně plnily svůj účel. Je doba, kdy máme prostředky a technologie, které nám ulehčí práci. Dokud ovšem naše mysl zůstane na místě a nebude se rozvíjet, můžeme mít sebelepší technologie, ale nikam nepostoupíme. My sami si bráníme dostat se o krok vpřed. Takže i když se vyvíjí technologie, musíme se i my sami neustále vyvíjet a své myšlení této technologii přizpůsobovat. Ne ve smyslu stát se otroky technologie, ale dokázat ji co nejefektivněji využívat.

7 Seznam použité literatury

Knihy:

[1] Carda Antonín, Kunstová Renáta. Workflow Procesní řízení. První vydání. Praha: Grada Publishing, spol s.r.o, 2001. Počet stran 136. ISBN 80-247-0200-2

[2] Fišer Roman, Procesní řízení pro manažera. První vydání, Praha: Grada Publishining, a.s., 2014. Počet stran 176. ISBN 978-80-247-5038-5

[3] Emmett Stuart, Řízení zásob. První vydání. Brno: Computer Press, a.s. 2008. Počet stran 298. ISBN 978-80-251-1828-3

[4] Mačát Václav, Sixta Josef, Logistika Teorie a praxe. První vydání. Brno: CP books, a.s. 2005. Počet stran 315. ISBN 80-251-0537-3

[5] Bigoš Peter, Kiss Imrich, Ritók Juraj, Materiálové toky a Logistika. Druhé vydání. Košice: Technická univerzita, Strojnícka fakulta 2008. Počet stran 156. ISBN 978-80-553-0129-7

[6] Jurová Marie a kolektiv. Výrobní a logistické procesy v podnikání. První vydání. Praha: Grada Publishing, a.s. 2016. Počet stran 264. ISBN 978-80-247-5717-9

[7] Arlow Jim, Neustadt Ila, UML2 a unifikovaný proces vývoje aplikací. Dotisk prvního vydání, Brno: Computer Press,a.s 2011. Počet stran 567. ISBN 978-80-251-1503-9

[8] Buchalcevová Alena, Metodiky a vývoje a údržby informačních systémů. První vydání. Praha: Grada Publishing, a.s. 2005. Počet stran 164. ISBN 80-247-1075-7

[9] Duben Jiří, Objektové modely podniku. První vydání. Praha: Grada Publishing, spol. s r.o. 1996. Počet stran 200 ISBN 80-7169-251-6

[10] Dlouhý Martin, Fábry Jan, Kuncová Martina, Hladík Tomáš, Simulace podnikových procesů. První vydání. Brno Computer Press, a.s. 2007. Počet stran 202. ISBN 978-80-251-1649-4

[11] Kraval Ilja, Analytické modelování informačních systémů pomocí UMl v praxi. První vydání. Lipina: Object Consulting 2010. Počet stran 140. ISBN 978-80-254-6986-6

Články v časopisech:

[12] Nisler Jan, *Důležitost dokumentace procesů pro automobilový průmysl*. SystemOnline. 2016 [online] [cit 2016-11-12] Dostupné z: <https://www.systemonline.cz/bpm-procesni-rizeni/dulezitost-dokumentace-procesu-pro-automobilovy-prumysl.htm>

[13] Nisler Jan, Systém je jako stavba. OR-CZ strana 20. 2014 [online] [cit 2016-11-08] Dostupné z: <http://www.orcz.cz/ke_stazeni/casopis_2014_or-system.pdf>

[14] Nisler Jan, Realita vývoje systému očima studenta. OR-CZ strana 21. 2014 [online] [cit 2016-11-08] Dostupné z: <http://www.orcz.cz/ke_stazeni/casopis_2014_or-system.pdf>

Odborné články

[15] Küng Stefan, Onken Lübbe, Large Simon, TortoiseSVN: Subversion klien pro Windows, Datum vydání 2013-11-11. [online] [cit 2016-11-15] Dostupné z: <https://sourceforge.net/projects/tortoisesvn/files/OldFiles/1.8.5/Documentation/TortoiseSVN-1.8.5-zh_CN.pdf/download>

[16] Douglas K Barry, Business Process Modeling Initiative (BPMI.org) 2005 [online] [cit 2016-12-4] <http://www.service-architecture.com/articles/web-services/business_process_modeling_initiative_bpmi.org.html >

[17] Kolektiv autorů, BPMI, Business Process Modeling Notation, 2002 [online] [cit 2016-12-05] <http://xml.coverpages.org/BPMN-NWG200211.pdf>

[18] Alaaeddine Yousfi, Rajaa Saidi, Anind K. Dey ,Variability patterns for business processes in BPMN, 2015 [online] [cit 2016-12-06] dostupné z: < https://www.springerprofessional.de/variability-patterns-for-business-processes-in-bpmn/11692396 >

[19] Peter Y. H. Wong, Jeremy Gibbons, A Relative Timed Semantics for BPMN, 2009 [online] [cit 2016-12-07] dostupné z: <http://www.cs.ox.ac.uk/jeremy.gibbons/publications/relative.pdf>

[20] Assaf Arkin, Business Process Modeling Language, 2002 [online] [cit 2016-12-07] dostupné z: < http://xml.coverpages.org/BPML-2002.pdf >

[21] Jean-Jacques Dubray, Eigner, A Novel Approach for Modeling Business Process Definitions [online] [cit 2016-12-07] dostupné z: < http://www.ebpml.org/ebpml2.2.doc >

[22] OMG, Business Process Modeling Notation (BPMN), 2004 [online] [cit 2016-12-07] dostupné z: < http://bpms.ru/fileadmin/pdf/bpmn-1.0.pdf >

[23] Van der Aalst, ter Hofstede, Kiepuszewski,Barros, Workflow patterns, 2002 [online] [cit 2016-12-07] dostupné z: < http://www.workflowpatterns.com/documentation/documents/wfs-pat-2002.pdf >

[24] Van der Aalst, ter Hofstede, Kiepuszewski,Barros, Advanced Workflow patterns [online] [cit 2016-12-07] dostupné z: < http://martinfowler.workflowpatterns.com/documentation/documents/coopis.pdf >

[25] White,, BPMN: Past, Present, and Future, 2012 [online] [cit 2016-12-07] dostupné z: < http://bpm2012.ut.ee/wp-content/uploads/2012/03/keynote-white.pdf >

[26] OMG, Business Process Model and Notation (BPMN) 2.0, 2011 [online] [cit 2016-12-07] dostupné z: < http://www.omg.org/spec/BPMN/2.0/ >

[27] OMG, Business Process Model and Notation (BPMN) 2.0.2, 2014 [online] [cit 2016-12-07] dostupné z: < http://www.omg.org/spec/BPMN/2.0.2/ >

[28] Arevalo, Escalona, Ramos, Dominguez-Munoz, A metamodel to integrate business processes time perspective in BPMN 2.0, 2016 [online] [cit 2016-12-07] dostupné z: < http://www.sciencedirect.com/science/article/pii/S0950584916300842 >

[29] Anacleto Correiaa, Fernando Brito e Abreu, Adding Preciseness to BPMN Models, 2012 [online] [cit 2016-12-07] dostupné z: < http://www.sciencedirect.com/science/article/pii/S2212017312004768 >

[30] Marlon Dumas, Alexander Grosskopf, Thomas Hettel, Moe Wynn, Semantics of Standard Process Models with OR-Joins

[31] Carlos Figueira, David Aveiro, A New Action Rule Syntax for Demo Models Based Automatic worKflow procEss geneRation, 2014 [online] [cit 2016-12-07] dostupné z: < https://link.springer.com/chapter/10.1007/978-3-319-06505-2_4 >

[32] Michael zur Muehlen, Jan Recker, Marta Indulska, Sometimes Less is More: Are Process Modeling Languages Overly Complex?, 2008 [online] [cit 2016-12-07] dostupné z: < http://www.academia.edu/2753950/Sometimes_less_is_more_Are_process_modeling_languages_overly_complex >

[33] Ralf Laue, Wilhelm Koop,and Volker Gruhn, Indicators for Open Issues in Business Process Models, 2016 [online] [cit 2016-12-07] dostupné z: < https://link.springer.com/chapter/10.1007/978-3-319-30282-9_7 >

[34] Peter Y.H. Wong, Jeremy Gibbons, Formalisations and applications of BPMN, 2009 [online] [cit 2016-12-07] dostupné z: < http://www.sciencedirect.com/science/article/pii/S0167642309001282 >

[35] Michele Chinosi, Alberto Trombetta, BPMN: An introduction to the standard, 2011 [online] [cit 2016-12-07] dostupné z: < https://www.journals.elsevier.com/computer-standards-and-interfaces/most-cited-articles >

[36] Felix Kossak, Christa Illibauer, and Verena Geist, Event-Based Gateways: Open Questions and Inconsistencies, 2012 [online] [cit 2016-12-07]

dostupné z: < https://link.springer.com/chapter/10.1007%2F978-3-642-33155-8_5 >

[37] Mateja Kocbek, Gregor Jošt, Marjan Heričko, and Gregor Polančič, Business Process Model and Notation: The Current State of Affairs, 2015 [online] [cit 2016-12-07] dostupné z: < http://www.comsis.org/archive.php?show=ppr510-1406 >

[38] Abel Armas-Cervantes, PaoloBaldan, MarlonDumas, Luciano Garcia-Bañuelos, Diagnosing behavioral differences between business process models: An approach based on event structures, 2015 [online] [cit 2016-12-07] dostupné z: < https://www.researchgate.net/publication/284012770_Diagnosing_behavioral_differences_between_business_process_models_An_approach_based_on_event_structures >

[39] Ahmed Kheldoun, Kamel Barkaoui, and Malika Ioualalen, Specification and Verification of Complex Business Processes - A High-Level Petri Net-Based Approach, 2015 [online] [cit 2016-12-07] dostupné z: < https://link.springer.com/chapter/10.1007/978-3-319-23063-4_4 >

Webové stránky

[40] odkaz na webové stránky skupiny OMG http://www.omg.org/

[41] odkaz na workflow patterns http://mgx.com.pl/WP/WorkFlowPaternsBPMN2d_WP28_-_Blocking_Discriminator_2.htm

[42] OMG, BPMN 1.2, 2009

[43] OMG, Business Process Modeling Notation, odkaz: http://www.omg.org/spec/BPMN/1.2/

[44] Odkaz na standart : ISO/IEC 19510:2013

8 Přílohy

1) Obrázek všech typů eventů

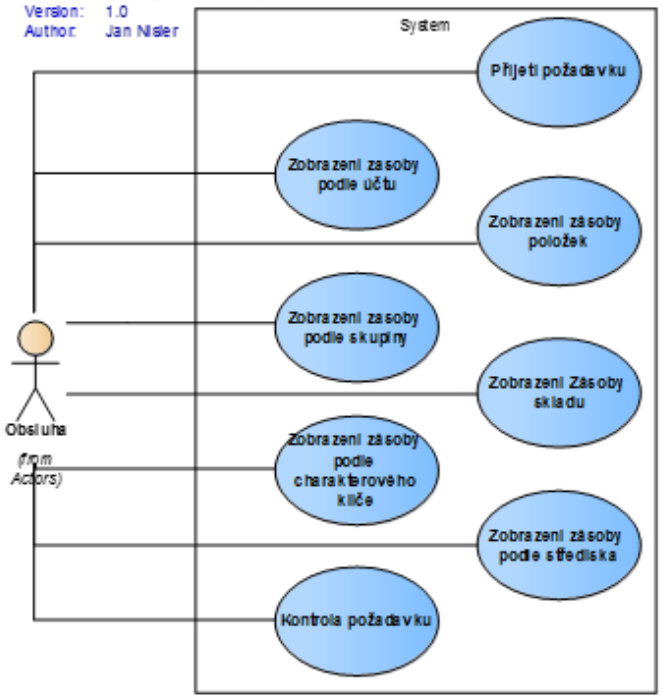
Type	Start			Intermediate				End
	Normal	Event Subprocess	Event Subprocess non-interrupt	catch	boundary	boundary non-interrupt	throw	
None								
Message								
Timer								
Conditional								
Link								
Signal								
Error								
Escalation								
Termination								
Compensation								
Cancel								
Multiple								
Multiple Parallel								

2) Část Dokumentace:

UCDiagramy_Spravy_Zasob

Use_Case_Spravy_Zasob

Name: Use_Case_Spravy_Zasob
Package: UCDiagramy_Spravy_Zasob
Version: 1.0
Author: Jan Nisler



Obrázek 1

Boundary: **System**

Actor: **Obsluha**

UseCase: **Kontrola požadavku**

kontroluje se požadavek na spravu zásob

UseCase: **Přijetí požadavku**

Přijetí došlého požadavku na objednání produktu

UseCase: **Zobrazení Zásoby skladu**

Případ užití poskytuje obsluze veškeré informace o zásobách skladu. Konkrétně se jedná o zásoby
- Položek
- Šarží
- Výrobních čísel
- Paletových jednotek
- Balících jednotek

Basic Path: **Basic Path**

1. Obsluha v hlavním menu systému zvolí úlohu Zobrazení Zásoby skladu
2. Zobrazí se seznam všech skladů, na kterých je evidována zásoba

3. Obsluha nastaví kurzor na sklad a zvolí příslušnou funkci dle toho, jaké informace o zásobě skladu chce získat - viz alternativní scénáře
4. Obsluha ukončí úlohu
Alternate 4a Funkce Detail zásoby skladu - návrat k bodu scénáře 3
Alternate 4b Funkce Zásoba položek na skladě - návrat k bodu scénáře 3
Alternate 4c Funkce Zásoba šarží na skladě - návrat k bodu scénáře 3
Alternate 4d Funkce Zásoba výrobních čísel na skladě - návrat k bodu scénáře 3
Alternate 4e Funkce Zásoba palet na skladě - návrat k bodu scénáře 3
Alternate 4f Funkce Zásoba Balících jednotek na skladě - návrat k bodu scénáře 3
Alternate 4g Funkce Zásoba balíků na skladě - návrat k bodu scénáře 3
5. Nastaví se aplikace do stavu před spuštěním Use Case

Alternate: **Funkce Zásoba výrobních čísel na skladě**
1. Obsluha zvolí funkci Výrobní čísla položky na skladě
2. Zobrazí se v řádkovém formuláři seznam všech výrobních čísel položky

Alternate: **Funkce**

Alternate: **Funkce Zásoba šarží na skladě**
1. Obsluha zvolí funkci Šarže na skladě
2. Zobrazí se v řádkovém formuláři seznam všech šarží na skladě

Alternate: **Funkce Zásoba položek na skladě**
1. Obsluha zvolí funkci Položky skladu
2. Zobrazí se v řádkovém formuláři seznam zásob všech položek skladu

Alternate: **Funkce Zásoba palet na skladě**
1. Obsluha zvolí funkci Palety na skladě
2. Zobrazí se v řádkovém formuláři seznam všech palet na skladě

Alternate: **Funkce Zásoba Balících jednotek na skladě**
1. Obsluha zvolí funkci Balící jednotky na skladě
2. Zobrazí se v řádkovém formuláři seznam všech balících jednotek na skladě

Alternate: **Funkce Zásoba balíků na skladě**
1. Obsluha zvolí funkci Balíky na skladě
2. Zobrazí se v řádkovém formuláři seznam všech balíků na skladě

Alternate: **Funkce Zkouška**

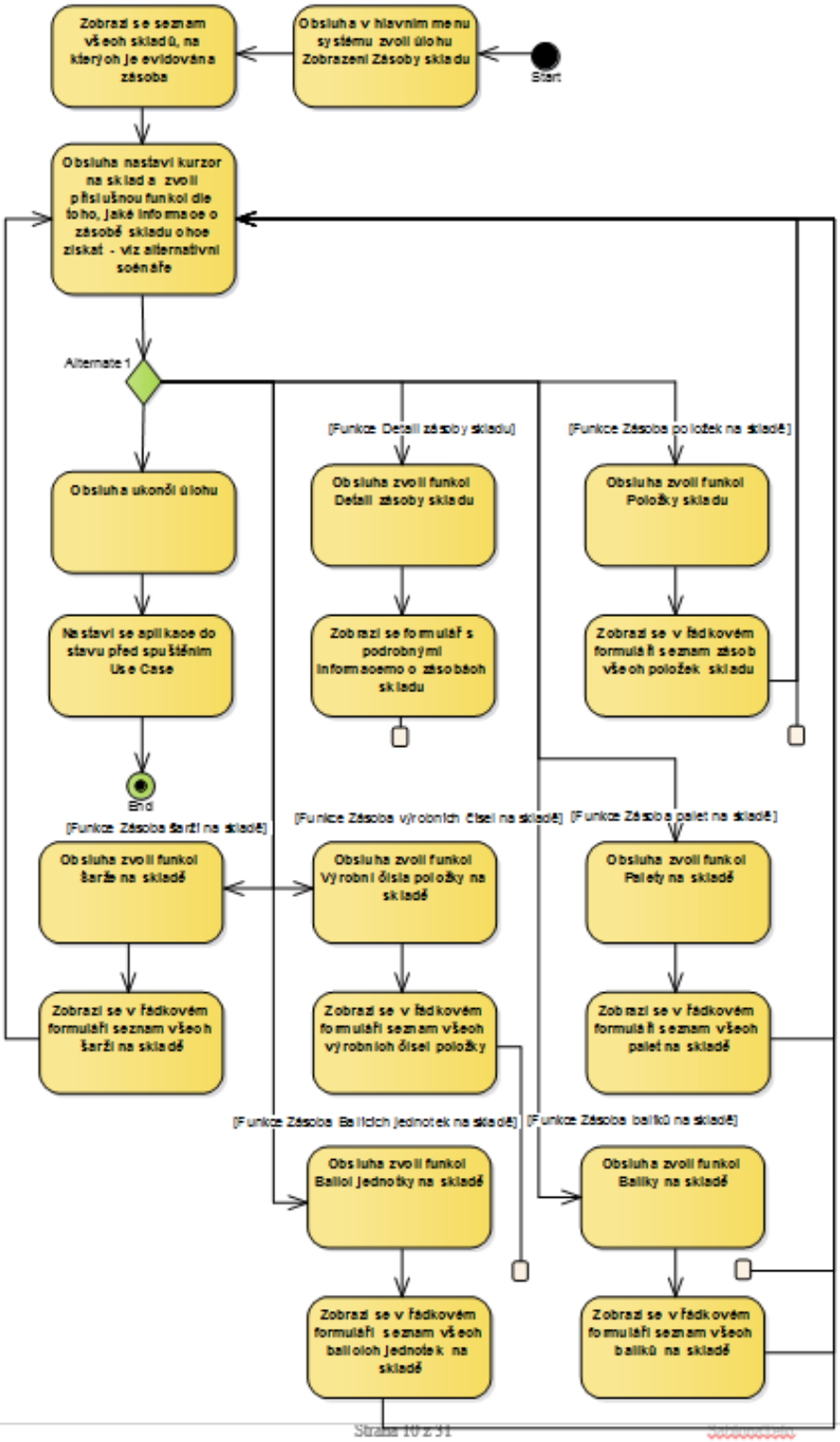
Alternate: **Funkce Detail zásoby skladu**
1. Obsluha zvolí funkci Detail zásoby skladu
2. Zobrazí se formulář s podrobnými informacemo o zásobách skladu

Activity: **Zobrazení Zásoby skladu_ActivityGraph**

Zobrazení Zásoby skladu_ActivityGraph

Name: Zobrazení Zásoby skladu_ActivityGraph
Package: UCDiagramy_Spravy_Zasob
Version: 1.0
Author: Jan Nisler

Obrázek 2



3) Šablona dokumentace:

Diagramy_Skladovych_Transakci

Autor: The Administrator

Datum: 31. 3. 2017

Verze: 1.2

Table of Contents

model >

Slovník

glossary >

{ModelGlossary.Term}

{ModelGlossary.Meaning}

< glossary

< model

package >

{Pkg.Name}

{Pkg.Notes}
package element >
linked document >
< linked document
< package element
diagram >

{Diagram.Name}

{Diagram.DiagramImg}
Obrazek {Diagram.Figure}

element >

{Element.Type}: ***{Element.Name}***

{Element.Notes}
< element
< diagram
element >

{Element.Type}: ***{Element.Name}***

{Element.Notes}
scenario >

{ElemScenario.Type}: **{ElemScenario.Scenario}**
structured scenarios >

{Scenario_Structured.Step}. {Scenario_Structured.Action}
exception >

{Exception.Type} {Exception.Step} {Exception.Name} - návrat k bodu scénáře {Exception.Join}
< exception
< structured scenarios

< scenario
attribute >

Atribut **{Att.Name}**

{Att.Notes}
< attribute
diagram >
< diagram
child elements >
< child elements
< element
child package >
< child package
< package

Oskenované zadání práce

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2016/2017

Studijní program: Systémové inženýrství a informatika
Forma: Prezenční
Obor/komb.: Informační management (im2-p)

Podklad pro zadání DIPLOMOVÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Nisler Jan	Zahradnická 37, Moravská Třebová - Předměstí	I1500740

TÉMA ČESKY:

Grafické mapování systému

TÉMA ANGLICKY:

Graphical mapping of system

VEDOUCÍ PRÁCE:

doc. Ing. Hana Tomášková, Ph.D. - KIT

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl Práce:
Vymodelovat do grafického prostředí skladové hospodářství ve firmě zabývající se vývojem ERP. Definovat jak se bude přetvářet textový popis do grafické podoby. Následně generovat celkovou dokumentaci z grafického prostředí.

1. Úvod
2. Cíl práce
2.1. Teoretická část
2.2. Praktická část
3.Teoretická část
3.1. Skladové hospodářství
3.2. Syntaxe BPMN
3.3. Syntaxe UML - Use Case
3.4. Enterprise Architect v.12
4. Praktická část
4.1. Procesy Skladového hospodářství
4.2. Typové úlohy Skladového hospodářství
5. Souhrn, vyhodnocení a přínos
6. Závěr
7. Zdroje
8. Přílohy

SEZNAM DOPORUČENÉ LITERATURY:

Arlow, Jim, Neustadt, Ila. UML a unifikovaný proces vývoje aplikací, Brno vydání První: Copyright ? Computer Press, rok 2003. počet stran 387. ISBN 80-7226-947-X

Schmuller, Joseph, Název: Myslíme v jazyku UML, knihovna programátora: Místo Praha vydání První: Grada Publishing, spol. s.r.o., rok 2001. počet stran 360. ISBN 80-247-0029-8.

Podpis studenta: ..... Datum: 11.10.2016

Podpis vedoucího práce: ..... Datum: .....