Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra Informačních Technologií

Diagramy chování a jejich praktické použití napříč notacemi UML, SysML, BPMN

Diplomová práce

Autor práce: Bc. Vilém Procházka

Studijní obor: Aplikovaná Informatika

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

Hradec Králové 2022

Prohlášení

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

.....

Bc. Vilém Procházka

14. listopadu 2022

Poděkování

Touto částí chci poděkovat vedoucí mé práce za ochotu, trpělivost a pomoc při výběru tématu a jeho samotným psaním.

Anotace

Tato diplomová práce pojednává o analýze a praktickém použití diagramů chování. Jejich modelování bude provedeno napříč notacemi a modelovacími jazyky UML 2.0, BPMN 2.0 a SysML. Tyto notace, včetně jejich diagramů budou podrobněji popsány. Záměrem této práce je vytvoření diagramů chování v různých notacích, jejich popis a možný význam pro modelované business procesy. U diagramů, které jsou postaveny na podobných základech, proběhne následně popsání a porovnání podobností a rozdílů. Ústředním tématem pro modelování je elektromobilita, specifičtěji se u většiny diagramů bude jednat o modelování procesu vývoje nového modelu či prototypu elektrického vozidla. Na konci práce dojde ke shrnutí zjištěných poznatků při modelování a bude určen nejvhodnější typ diagramu pro co nejlepší znázornění daného obchodního procesu.

Klíčová slova: Modelovací jazyky, notace, BPMN, UML, SysML, diagramy, elektromobilita, analýza, obchodní proces

Anotation

Title: Behavioural diagrams and their use across the notations UML, SysML and BPMN

This diploma thesis discusses the analysis and practical use of behavior diagrams. Their modeling will be done across notations and modeling languages UML 2.0, BPMN 2.0 and SysML. These notations, including their diagrams, will be described in more detail. The purpose of this work is to create behavior diagrams in different notations, create their description and state possible significance for modeled business processes. For diagrams that are built on similar foundations, similarities and differences will be described and

compared. The central topic for modeling is electromobility, more specifically, most of the modeled diagrams will be about the development process of a new model or prototype of an electric vehicle. At the end of the work, the knowledge gained during modeling will be summarized and the most suitable type of diagram will be determined for the best representation of the given business process.

Keywords: Modeling languages, notations, BPMN, UML, SysML, diagrams, electromobility, analysis, business process

Obsah

Seznam obrázků

1 Úvod

Jako téma této diplomové práce byly vybrány Diagramy chování a jejich praktické použití napříč notacemi UML, SysML, BPMN. Motivem pro výběr tématu byla má přirozená zvídavost zevrubněji prozkoumat analytickou stránku informačních technologií. Konkrétněji naučit se o dalších modelovacích notacích či jazycích a jejich diagramech. Cílem práce bude modelování diagramů chování skrze zmíněné notace a jejich následný rozbor a u některých porovnání jejich podobností či rozdílů. V teoretické části je uvedeno něco o historii a vývoji modelovacích jazyků v průběhu několika posledních dekád. V téže kapitole jsou postupně popsány modelovací jazyky a notace zmíněné v názvu práce. Je popsán jejich důvod vzniku, principy, jež mají modelovat. Následují také popis diagramů, které do nich patří, jejich prvků a možnostech využití. Nejpodstatnější diagramy chování jsou z důvodu vyšší důležitosti vzhledem k tématu rozebrány do větších detailů. Praktická část práce se zabývá rozborem a popisem vytvořených diagramů chování, které budou modelovány v ohledu na téma elektromobility. To je ještě před diagramy samotnými krátce rozebráno. Z notací jsou vybrány určité diagramy, které mají ukázat různé možnosti zápisu chování v určité aktivitě či vybrané třídě. U diagramů, které si jsou napříč notacemi podobné, jsou popsány rozdíly v modelování, vylepšení vůči ostatním notacím a proč je vhodné diagram právě v tomto typu notace použít oproti dalším probíraným notacím či jazykům. Ve výsledcích je uvedeno shrnutí přínosu jednotlivých diagramů chování a je určeno, které z diagramů či notací se nejvíce hodí pro vybraný modelovaný proces.

2 Cíl práce

Cílem této diplomové práce je analýza diagramů chování, jejich využití na praktických příkladech, které se ponesou na tématu elektromobility, a analýza modelovacích notací, ve kterých budou diagramy vytvářeny. Těmito modelovacími notacemi a jazyky jsou UML 2.0, BPMN 2.0 a SysML. Bude představena historie modelovacích jazyků a vznik jednotlivých zmíněných modelovacích notací. Následně bude připraven popis principů a použití jednotlivých notací, diagramů do nich patřících, jak mohou více pomoci pro správné pochopení modelovaných situací a dále také, co přináší nového nebo jiného do oblasti modelování. Samotné diagramy chování budou rozebrány do větších detailů ve vlastní kapitole. Na základě těchto poznatků bude připraven a představen business proces na aktuální téma elektromobility a vývoje elektromobilů, který bude následně ve zmíněných notacích vymodelován a vytvořené diagramy budou popsány. V rámci diagramů, které jsou postaveny na podobném principu, nastane porovnání a zhodnocení jejich rozdílů a výhod.

3 Modelovací notace

Mnohdy se říká, že obrázek bývá lepší, nežli tisíce slov a toto tvrzení neplatí o nic méně v informačních technologiích. Proto bývají vizualizace procesů, a to především během vývojových fází aplikací a systémů, zásadní pro správnou týmovou spolupráci a komunikaci mezi jednotlivými členy týmu. Modelování bývá používáno zejména v softwarovém inženýrství nebo pro znázornění business procesů a lze říci, že existují dva hlavní způsoby, kterými se diagramy používají jako součást takovýchto procesů. prvním způsobem je dopředný design (dopředné inženýrství), kde se modelování a návrh provádí před samotným tvořením kódu softwarové aplikace. Tento způsob bývá nejčastější a obvykle se používá k tomu, aby programátorům pomohl lépe vidět systém, který se pokoušejí vyrobit. Druhou variantou je zpětný design. Zde se vytváření modelu provádí až po napsání kódu a diagramy tak následně fungují jako dokumentace pracovního postupu daného projektu. To pomáhá vývojářům a manažerům pohlédnout na proběhlý vývoj projektu tak, jak proběhl ve skutečnosti. To může vést například k poučení z provedených chyb nebo naopak vyzdvihnutí určitých praktik během tvorby systému, aby vývoje budoucích projektů mohly proběhnout hladčeji. Nicméně, ať už jsou diagramy použity jedním nebo druhým způsobem, poskytují způsob, jakým lze vizualizovat velkou škálu aspektů projektu, a to včetně toho, kdo má být zodpovědný za jakou činnost. Odvětví softwarového vývoje využívalo již od svého počátku před desítkami let mnoho různých způsobů, jak modelovat a vizualizovat vývoj. V roce 1997 byl však vydán průmyslový standard pro modelování softwaru zvaný Unified Modeling Language, který se v novějších iteracích používá doteď a je jedním z nejsilnějších a nejdůležitějších nástrojů pro softwarové inženýrství.[2]

3.1 Cesta ke vzniku UML

Jak již bylo řečeno, v průběhu vývoje informačních technologií vzniklo poměrně velké množství různých metod a notací. Pro příklad to mohou být metody či notace pro návrh, strukturu a zpracování nebo ukládání informací. S většinou již dříve vynalezených popsaných pojmů je možné se běžně setkat také v dnešní době, jelikož jsou, jako například dědičnost či polymorfismus, základními stavebními kameny objektově orientovaného programování. V průběhu 70. letech však bylo programování bráno poměrně individuálně, což začalo být

velmi problematické s tím, jak narůstaly požadavky na vývoj, složitost a údržbu používaných systémů. Výsledkem toho nastalo to, co se dá nazvat softwarovou krizí.[3]

Pod tímto pojmem si lze představit moment, kdy již není možné psát počítačové programy efektivně a v rámci časových harmonogramů. Vzrůstající poptávka po nových typech softwarů a jejich složitosti se špatně slučovala s používáním starých nástrojů a neměnících se metod, které již začaly být tou dobou nedostačující. Nastaly problémy s kvalitou, efektivitou, správou a také s rozpočtem vyvíjených programů. [4]

Tato krize vedla ke vzniku principu strukturovaného programování, tedy více systematickému, disciplinovanému a kvantifikovatelnému přístupu k vývoji softwaru. Byly vytvořeny metody pro lepší strukturování systémů a pro procesy návrhu, vývoje a jejich údržby.Procesově orientované přístupy, jako kupříkladu HIPO (Hierarchy Input Processing Output) měly za hlavní cíl funkčnost systémů. Tato metoda pomohla celý systém rozdělit na menší části díky funkčnímu rozkladu. Každý prvek hierarchickém diagramu je popsán pomocí schématu "vstup-proces-výstup". Ve stejné době byly vyvinuty také přístupy, které se orientovaly na datovou strukturu. Příkladem toho může být Jacksonova metoda, ve které je struktura samotného programu odvozena od datových struktur zobrazených v grafické podobě. Zjednodušeně si lze představit dva sloupce, kde v prvním je struktura souboru dat a ve druhém již struktura programu (procesy, transakce) z datové struktury odvozená. Příkladem programovacího jazyku využívající tyto přístupy byl například COBOL. [3]

S roustoucí komplexností systémů však již nebylo vhodné tvořit programy "od nuly"a bylo nutné mít jistou míru udržovatelnosti a znovupoužitelnosti. S příchodem objektově orientovaných jazyků přišly také objektově orientované modelovací notace/jazyky. Vzhledem k množství objektově orientovaných metod a návrhových forem bylo prakticky nezbytné zavést unifikovaný modelovací jazyk (UML). Na počátku 90. let minulého století byly při programování již běžně používány objektově orientované metody, proto v roce 1994 začala firma Rational Software Corporation (později součást IBM) vytvářet první UML. Bylo rozhodnuto, že standardizovat se bude modelovací jazyk/notace, nikoliv metody samotné. Při vytváření notace byla integrována Boochova metoda, technika objektového modelování (OMT, dalo by se nazvat jako předchůdce UML) a objektově orientovaného softwarového inženýrství (OOSE) společně s dalšími prvky a metodami a výsledná notace byla nazvána jako UML 0.9. Cílem nebylo vytvoření úplně nové notace, ale přizpůsobení, rozšíření a zjed-

nodušení již existujících typů diagramů některých objektově orientovaných metod. Mezi ně patřily diagramy tříd, Jacobsonovy diagramy případů užití nebo Harelovy diagramy stavu. Způsoby zobrazení v minulosti použité u strukturovaných metod byly aplikovány také na UML. [3]

3.2 UML 2

V roce 1997 bylo vydáno UML ve verzi 1.1. Jako první byla tato verze přijata konsorciem Object Management Group (OMG) jako standard pro své členy, mezi které patří/patřily například IBM, Apple Computer nebo Hawlett-Packard. S myšlenkou na možnost interoperability bylo zajištěno, že se UML stane v dalších letech společně sdíleným vizuálním jazykem. [5]

Velkou aktualizace dostalo UML v září 2001 s verzí 1.4. Došlo k obohacení UML díky několika rozšířením a do diagramů byly přidány viditelnosti, artefakty a stereotypy. Další milník pro UML nastal na počátku roku 2005, kdy byl tato modelovací notace ve verzi 1.4.2 uznána jako standard Mezinárodní organizací pro normalizaci (ISO). V sprnu téhož roku bylo UML vydáno ve verzi 2.0, ve které došlo k přidání poměrně hodně nových prvků. Nejpodstatnější vylepšení v této verzi bylo přidání nových diagramů - objektový, balíčkový, diagram časování, či diagramy interakcí. Diagramy aktivit a sekvenční obdržely update v podobě nových funkcí. Diagram spolupráce byl přejmenován na diagram komunikace a do dalších stávajících diagramů bylo přidáno nekolik nových funkcí a dalších změn. V následujících několika letech byly prováděny spíše drobnější změny, ty větší přišly až s verzí 2.4.1. Byla revidována stávající verze 2.3, ze které byly vylepšeny určité prvky a byly provedeny změny tříd, balíčků a stereotypů. Zatím poslední větší změnou prošlo UML v roce 2015, kdy byla aktualizováno na verzi 2.5. UML jako takové bylo "zjednodušeno". Došlo k zavedení rychlejšího a efektivnějšího generování modelů, odstranění zastaralých funkcí a některých pomocných konstruktů, jako například šablon. Nejnovější verzí UML je 2.5.1 a byla zavedena v prosinci roku 2017. [6]

Diagramy v UML lze jednoduše rozdělit na 3 kategorie. Jsou to strukturální diagramy, diagramy chování a diagramy interakcí.

3.2.1 Strukturální diagramy

Strukturální diagramy se používají pro reprezentaci statického pohledu na modelovaný systém. Jak název vypovídá, těmito diagramy je představována část systému, která tvoří jeho strukturu a ukazují na různé objekty v něm.[7]

Mezi strukturální diagramy patří:

- Diagram tříd - znázorňuje třídy použité v daném systému, jejich vzájemné vztahy a "spolupráci", a také užíváné metody a atributy. Mimo jiné může obsahovat také rozhraní či balíčky [7]
- Objektový diagram - je podobný diagramu tříd, ale je soustředěn na již vytvořené objekty, jinými slovy je to tedy diagram instancí. Jednotlivé objekty obsahují svá data, tedy hodnoty, jimiž jsou naplněny. Ukazuje podrobnější interkace mzi objekty v určitém okmažkiku během chodu systému [7]
- Balíčkový diagram - ukazuje uspořádání a organizaci prvků modelu v daném projektu. V tomto diagramu je možné zobrazit jak strukturu, tak závislosti mezi jednotlivými subsystémy nebo moduly, což pomáhá například ukázat určitou aplikaci jako vícevrstvou z více úhlů pohledu [8]
- Komponentový diagram - používá se při modelovní fyzických aspektů objektově orientovaných systémů používaných pro vizualizaci, specifikaci a dokumentaci komponentových systémů a také pro konstrukci spustitelných systémů prostřednictvím dopředného a zpětného inženýrství. [9]
- Diagram nasazení - je používán pro vizualizaci fyzického hardwaru a softwaru daného systému. Ukazuje konfiguraci správy run-timových uzlů a komponent, které se na nich nachází [10]

3.2.2 Diagramy chování

Jakýkoliv systém nacházející se v reálném světě může být reprezentován buďto ve statické nebo dynamické ("pohyblivé") formě. Říká se, že systém není kompletní, pokud není vyjádřen jak statickým, tak dynamickým způsobem. Díky tomu lze obecně říci, že diagramy chování představují fungování celého systému. [6]

Patří mezi ně:

- Diagram aktivit
- Diagram případů užití (UC)
- Stavový diagram (state machine diagram)

Tyto typy diagramů budou detailněji popsány v jedné z následujících kapitol

3.2.3 Diagramy interakcí

Diagramy interakcí nejsou ve své podstatě nic jiného, než určitá podmnožina diagramů chování. Jsou používány k vizualizaci toku mezi různými prvky pocházejících z případů užití modelovaného systému. Je v nich ukazováno, jak probíhá interakce, komunikace či jak proudí data mezi dvěma a více entitami. [6]

Diagramy patřící do této kategorie jsou:

- Diagram časování - jsou používány k zobrazení interakcí, kde je primárním účelem úvaha o čase při průchodu systémem. Zaměřují se na měnící se podmínky v rámci takzvaných lifeline (jednotliví účastníci interakcí) na lineární časové ose. Popisují chování jednotlivých účastníků a jejich interakcí a zaměřují se na časy událostí způsobujících změny v modelu [11]
- Sekvenční diagram - modeluje spolupráci objektů na základě časové posloupnosti. Je v něm ukázáno, jak spolu objekty navzájem iteragují v rámci konkrétního scénáře v případech užití. Objekty zasílají zprávy jiným objektům v rámci požadavků na vykonání určitých akcí, operací a dotazů. [12]
- Diagram komunikace (spolupráce) - je rozšířením objektového diagramu, s tím rozdílem, že ukazuje nejen asociace mezi jednotlivými objekty, ale také zprávy, které si objetky mezi sebou posílají [13]

3.3 BPMN 2.0

Business Process Model and Notation (BPMN) je grafickou notací, jež modeluje kroky plánovaného business procesu od počátku až do konce. Vizuálně zobrazuje podrobnou sekvenci

jednotlivých business aktivit, a informačních/datových toků, které jsou potřebné pro kompletní znázornění daného procesu. Účelem BPMN je modelovat způsoby, kterými je možné zlepšit efektivitu, brát v úvahu nové, či ne úplně očekávané události nebo najít možnosti, jakými získat určitě konkurenční výhody. [14]

BPMN bylo vyvinuto konsorciem Business Process Management Initiative (BPMI) v roce 2004, kdy vydalo jeho verzi 1.0 a od té doby již prošlo množstvím revizí. BPMI se v roce 2005 sloučilo se skupinou Object Management Group, která následně převzala iniciativu nad dalším vývojem notace. V roce 2011 vydalo OMG verzi 2.0 a přešlo se od původního názvu Business Process Modeling Notation na Business Process Model and Notation, zkratka BPMN zůstala zachována. Důvodem pro tuto změnu byl fakt, že se již jednalo o více než pouhou notaci. Verze 2.0 přinesla podrobnější standard pro modelování business procesů s použitím bohatší sady symbolů a zápisů pro diagramy obchodních procesů (Business Process Diagrams). V roce 2013 bylo BPMN uznáno jako standard ISO. Zatím poslední verze má označení 2.0.2 a díky ní bylo BPMN doplněno o metodu rozhodovacího vývojového diagramu zvanou Decision Model and Notation Standard. Důvodem k tomuto rozhodnutí byl fakt, že samotné BPMN jako takové se pro navrhování rozhodovacích toků příliš nehodilo.[14]

Hlavním cílem při navrhování BPMN byla myšlenka, aby byla poskytnuta notace snadno srozumitelná všem obchodním partnerům. Tím je myšleno od business analytiků, přes technické vývojáře a pro lidi, kteří budou firemní procesy řídit a monitorovat, až k business manažerům a dalším lidem na výše postavených pozicích. Z obecného hlediska lze BPMN použít jako pomoc pro dosažení společného žádaného cíle všech zúčastněných stran. To vše v ohledu k projektu přijímajícího jeden společný jazyk pro popis požadovaných business procesů. Z detailnějšího a internějšího úhlu pohledu je BPMN mířeno na lidi, kteří budou tyto procesy implementovat a dodává dostatek podrobností, aby bylo možné dosáhnout co nejpřesnějšího výsledku. V tom nejideálnějším případě dovede překlenout propast mezi záměrem, co měl proces dělat, a následnou implementací. To pomáhá vyhnout se komunikačním mezerám a "hluchým"místům, které by jinak mohly v projektech vzniknout, pokud by nebyl vývojářům poskytnut přesný náhled do posloupnosti obchodních aktivit. Další výhodou používání diagramů je pomoc při vytváření XML dokumentů potřebných k prová-

dění některých procesů. Hlavní XML standard je zván BPEL nebo BEPEL4WS (Business Process Execution Language for Web Services).[14]

Jak již bylo řečeno, diagramy v BPMN se označují jako Business Process Diagrams (BPD) a jsou vždy skládány z několika grafických prvků. Ty se dělí do čtyř kategorií - jsou to tokové objekty (Flow Objects), spojovací objekty (Connecting Objects), takzvané plavecké dráhy (Swimlanes) a artefakty (Artifacts). Jako samostatnou kategorii by bylo možné také označit data, nicméně ty se běžně řadí mezi artefakty. [14]

Tokové objekty formulují chování business procesů.

- Události (Events) - něco, co se odehrálo během průchodu business procesem, reprezentovány formou kruhu. Je jimi ovlivněn tok daného procesu, běžně mají svou příčinu (trigger) a nějaký důsledek. Dle umístění v procesu je lze rozdělit na počáteční - spouštěč procesu, existuje více druhů (none, message, timer, conditional, signal a další), průběžná událost - ovlivňuje průběh procesu a určuje jeho další směr, a událost koncová - tedy výsledek aktivity u daného business procesu. [14]
- Aktivity (Activities) - reprezentují akce, které je třeba vykonat. Obecně lze říci, že jde o nějakou práci, která má být udělána společností daný proces provádějící. Jsou zobrazovány pomocí obdélníku s kulatými rohy. Můžou to být buďto jednoduché úkoly (task), tedy aktivity v rámci procesu a podprocesy (subprocesy), které v sobě obsahují a skládají více drobnějších aktivit do jedné složené aktivity. Mají své vlastní samostatné počáteční a koncové události a sekvenční toky z nadřazeného procesu nesmí překročit jejich hranici. Dále sem lze zařadit transakce, což je typ subprocesu, u kterého musejí být všechny vnitřní aktivity brány jako jeden celek, tím pádem je nutné, aby vše uvnitř bylo splněno pro dosažení kýženého cíle. Pokud některá z nich selže, musí být všechny vráceny na počáteční stav (undone). Od běžných subprocesů se liší dvojitým ohraničením. Posledním typem je aktivita volání. Jedná se o bod v průběhu procesu, kde je znovu použit globální proces nebo globální úloha. Od ostatních typů aktivit se volání odlišuje tučným okrajem okolo oblasti aktivity. [14]
- Brány (Gateways) - určují rozvětvení či sloučení toků procesu (v závislosti na předem vyjádřených podmínkách. Bývají znázorněny kosočtvercem. Existuje několik různých typů bran, prvním z nich je exkluzivní, který se používá při vytvoření mnohdy i něko-

lika alternativních toků v procesu, ale lze se vydat pouze jednou z možných cest. Paralelní brány vytvářejí několik souběžných průchodů, ve kterých jsou aktivity plněny zároveň, není zde třeba žádných dalších podmínek. Inkluzivní varianta dává možnost průchodu branou více než jedním tokem najednou, nicméně nakonec všechny z nich zpravidla sloučí zpět do jednoho toku. Brány komplexního typu se využívají tam, kde jee potřeba modelovat složitější větvení toků nebo určité synchronizační chování. Brány existující na základě průběhu událostí (Event based) jsou využity tehdy, pokud může nastat jedna z více variant situací, na základě čehož je následně vybrán správný průchod. Exkluzivní Event based brána je velmi podobná klasické bráně na základě průběhu událostí. Vyhodnocuje se událost, aby se určilo, která ze vzájemně se vylučujících cest bude použita. Oproti tomu existuje paralelní Event based brána, kde na základě nějaké události dochází k průchodu dvěma či více toky a nedochází k žádnému vyhodnocení. [14]

Tokové objekty jsou mezi sebou spojovány spojovacími objekty a určují komunikaci a směr průchodu business procesem.

- Asociace - V diagramu zobrazena pomocí tečkované čáry. Používá se k prostému přidružení artefaktu nebo poznámkovému textu k objektu. Je možné na ni přidat šipku, která ukazuje směr. Pokud ukazuje na objekt, znamená to výsledek. Pokud jde šipka pryč od objektu na artefakt či text, ukazuje to, že byl přečten a zároveň aktualizován. V případě, že neobsahuje žádnou směrovost, znamená to, že je asociace spojena s jiným tokem, jež již udává směr. [14]

- Sekvenční tok - Modelován pomocí plné čáry se šipkou, udává pořadí, ve kterém se provádějí činnosti. Na začátku toku může být symbol kosočtverce, jež ukazuje některý z řady podmíněných toků vycházejících z aktivit, přičemž lomítko značí výchozí tok z rozhodnutí/aktivity. Spojované události, aktivity nebo brány nemohou přesahovat daný podproces či dráhu. [14]

- Tok zpráv - V modelu znázorněn coby přerušovaná čára, která má na začátku prázdný kruh a na konci nevyplněnou šipku. Udává, jaké zprávy mohou být předávány napříč hranice daných organizačních bloků (mezi bazény/pooly). Zpráva může mít pouze jednoho odesílatele a jednoho příjemce. Tento typ toku není možné používat k propojení aktivit či událostí v rámci jednoho poolu. [14]

Plavecké dráhy (swimlanes) jsou vizuálním mechanismem pro organizaci a kategorizaci aktivit založených na "cross-functional flowchart"diagramu a v BPMN se skládají ze dvou typů:

- Bazény (pools) - Jsou stěžejními účastníky procesu. Typicky oddělují nějaké organizační bloky. Bazény mohou obsahovat dráhy, kterými mohou být například oddělení ve firmách. Jejich funkce je „zapouzdřovat“ sekvenční toky mezi aktivitami v daném procesu. Mezi dráhami bazénu je možné pomocí sekvenčních toků překračovat, ne však mezi bazény samotnými. Jak bylo řečeno v jednom z předchozích bodů, pro tento účel existují toky zpráv. Bazén lze označit jako otevřený, pokud zobrazuje vnitřní detaily. V takovém případě obsahuje alespoň jednu dráhu a je zobrazen jako prázdný obdélník roztahující se na šířku nebo výšku celého diagramu. [15]

- Dráhy (lanes) - používají se k organizaci a kategorizaci aktivit v rámci poolu. Význam či smysl drah je čistě na uživateli modelujícím daný proces, BPMN jako takové jejich použití nespecifikuje. Dráhy jsou často používány pro znázornění takových věcí, jako jsou například interní role (manažer, vývojář, analytik), systémy (např. podniková aplikace) nebo interní oddělení (např. IT, finance, HR) a podobně. Dále také existuje možnost pruhy vnořovat do dalších pruhů. Například by mohla existovat vnější sada pruhů pro oddělení nějaké firmy a poté vnitřní sada pruhů pro role v rámci každého oddělení. Zobrazují se jako obdélník táhnoucí se na šířku nebo výšku bazénu. Dráha v sobě obsahuje objekty toku, spojovací objekty a artefakty. [16]

Artefakty umožňují vývojářům zasadit do modelu/diagramu nějaké informace s určitou přidanou hodnotou. Díky tomu je možné učinit daný model či diagram více čitelným.

- Data - jak již bylo zmíněno v jednom z předchozích odstavců, je možné je také vyhranit jako samostatnou kategorii. Nacházejí se zde 2 typy typy objektů, z nichž jedny jsou nazvány přímo Datové objekty. Ty není možné využívat v choreografiích, ale pouze v procesních diagramech. Dodávají informace, které aktivity se mají uskutečnit, či jaký má být jejich výsledek. Reprezentují buďto samotný objekt, či jejich kolekci. Druhým typem jsou datové sklady, které nabízejí způsob pro zisk či aktualizování uschovaných dat aktivitám, jež jdou mimo meze daného procesu. [14]

- Skupiny - jsou znázorněny obdélníkem se zaoblenými rohy a přerušovanými čarami. Skupina se používá k seskupení několika různých aktivit, ale neovlivňuje tok v diagramu. [14]

- Anotace - slouží k přesnějšímu popisu, aby na uživatele model, diagram nebo jeho části působily srozumitelnějším dojmem. [14]

Mezi další typy diagramů, které je mimo Business Process Diagram možné v BPMN modelovat, patří ještě diagram choreografie, kolaborační diagram a diagram konverzace.

- Diagram choreografie - choreografie je také typem procesu, ale liší se účelem a chováním od standardního procesu BPMN. Standardní proces nebo proces organizace je většině "modelářů"procesů známější a definuje tok aktivit konkrétní entity nebo organizace. Choreografie oproti tomu popisuje způsob, jakým účastníci business procesů koordinují své interakce. Důraz není kladen na koordinaci práce prováděné v rámci interakcí těchto účastníků, ale spíše na výměnu informací (tedy zpráv) mezi nimi.[17]

- Kolaborační diagram - zobrazuje interakce mezi dvěma či více podnikovými entitami nebo procesy. Kolaborace obvykle obsahuje dva nebo více bazénů, které představují účastníky spolupráce. Výměna zpráv mezi účastníky je znázorněna tokem zpráv, který spojuje dva bazény (nebo objekty ve bazénech). Zprávy spojené s toky zpráv se mohou v diagramu také objevit.V tomto typu diagramu mohou být využity všechny kombinace bazénů, procesů a choreografií. [18]

- Diagram konverzace - je konkrétní použití a neformální popis diagramu spolupráce. Obecně se jedná o zjednodušenou verzi kolaboračního diagramu, ale zároveň si tyto diagramy zachovávají všechny jeho funkce. Mohou se objevit zejména v diagramech účastníků (bazénů) konverzace, aby ukázaly, jak spolu konverzace a aktivity souvisejí. Konverzace jako taková je logické seskupení toku zpráv, které spolu mají určitý vztah. Tento vztah se v praxi často týká obchodních předmětů, které jsou předmětem zájmu firmy, např. „Objednávka“, „Zásilka a dodání“ a „Faktura“. Z toho důvodu bývá konverzace spojena se sadou párů ve formě název-hodnota, které jsou zaznamenány ve zprávách, jež jsou navzájem účastníky zasílány. Tímto způsobem může být zpráva směrována do konkrétní instance procesu zodpovědného za její příjem a zpracování. [19]

3.4 SysML

Systems Modeling Language (zkr. SysML) je obecný grafický modelovací jazyk pro specifikaci, analýzu, navrhování a ověřování různých systémů. Ty mohou zahrnovat hardware, software, firemní informace či data, personál, pracovní postupy a používaná zařízení. Konkrétně tento jazyk poskytuje grafické reprezentace se sémantickým základem pro modelování systémových požadavků, jejich chování, struktury a parametriky. Díky tomu se používají například i k integraci s jinými modely inženýrské analýzy. SysML bylo původně vyvinuto jako open source projekt a zahrnuje open source licenci pro jeho distribuci a použití. SysML je definováno jako rozšíření jazyka UML (Unified Modeling Language) pomocí mechanismu profilu UML. Toto rozšíření UML bylo navrženo tak, aby podporovalo činnosti systémového inženýrství.[20]

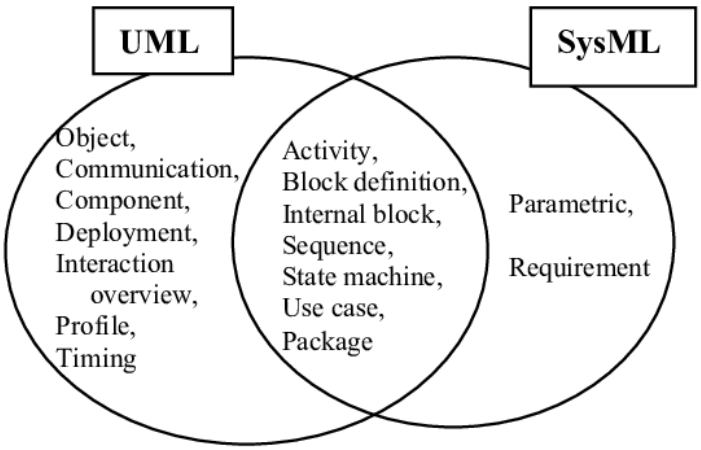
Dalo by se říci, že historie vývoje SysML byla na jeho počátku poměrně komplikovaná, vzhledem k počátečnímu konkurenčnímu boji mezi Iniciativa pro počátek vývoje SysML vznikla na začátku roku 2001 rozhodnutím skupiny International Council on Systems Engineering (INCOSE) Model Driven Systems Design upravit notaci UML pro potřeby návrhu aplikací systémového inženýrství. Po tomto úvodním činu INCOSE a Object Management Group (OMG), jež se stará o specifikaci UML, v červenci 2001 společně ustanovily OMG Systems Engineering Domain Special Interest Group (SE DSIG). SE DSIG s podporou INCOSE a ISO AP skupiny 233 vyvinulo požadavky na modelovací jazyk, které následně vydala OMG v UML for Systems Engineering Request for Proposal (UML pro SE RFP) v březnu 2003. V roce 2003 Cris Kobryn a Sanford Friedenthal zorganizovali a spolupředsedali konsorciu SysML Partners, neformálnímu sdružení předních výrobců a dodavatelů nástrojů, které zahájilo projekt specifikace open source s cílem vyvinout SysML v reakci na UML pro systémové inženýrství RFP. Původními technickými přispěvateli a spoluautory specifikace SysML 1.0a byli Laurent Balmelli, Conrad Bock, Rick Steiner, Alan Moore a Roger Burkhart. SysML Partners distribuovali své první návrhy specifikace SysML s otevřeným zdrojovým kódem v roce 2004 a v listopadu 2005 předložili SysML 1.0 před OMG k přijetí této technologie. Po sérii konkurenčních návrhů specifikace SysML byl v dubnu 2006 od OMG navržen tým pro sloučení SysML. Tento návrh byl odhlasován a přijat OMG v červenci 2006 jako OMG SysML, aby se odlišil od původní open source specifikace, od níž byla

notace odvozena. Protože je však OMG SysML odvozeno od open source SysML, zahrnuje díky tomu také open source licenci pro jeho distribuci a použití.[21]

Specifikace OMG SysML v. 1.0 byla vydána OMG jako dostupná specifikace v září 2007. Aktuální verze OMG SysML je v1.6, která byla vydána OMG v prosinci 2019. Kromě toho byl SysML publikován International Organisation for Standardization (ISO) v roce 2017 jako plnohodnotná mezinárodní norma (IS), ISO/IEC 19514:2017. OMG následně pracovala na další generaci SysML a 8. prosince 2017 podala žádost o návrh (Request for Proposals, RFP) pro verzi 2 v návaznosti na proces otevřené standardizace. Výsledná specifikace, která by měla zahrnovat vylepšení modelovacího jazyku ze zkušeností s jeho aplikací, bude obsahovat profil UML, metamodel a mapování mezi profilem a metamodelem. Druhá RFP pro SysML v2 Application Programming Interface (API) a služby RFP byla vydána v červnu 2018. Jejím cílem bylo zlepšit interoperabilitu nástrojů systémového inženýrství založených na modelu.[21]

3.4.1 Porovnání SysML a UML

SysML nabízí několik vylepšení specifických pro systémové inženýrství oproti UML, které bylo vyvinuto jako softwarový modelovací jazyk. Na obrázku 1 lze vidět, které modelovací prvky byly v SysML oproti UML zanechány, odebrány či přidány.



Obrázek 1: Množina prvků SysML vs. UML, Zdroj: [1]

Některé z vylepšení jsou zahrnuta v následujících bodech:

- V notaci SysML se lépe vyjadřují koncepty systémového inženýrství díky odstranění omezení v UML zaměřených na software a přidává dva nové typy diagramů. Těmi jsou diagramy požadavků a parametrické diagramy. První lze použít pro inženýrství požadavků; posledně jmenovaný oproti tomu může být použit pro analýzu výkonnosti a kvantitativní analýzu. V důsledku těchto vylepšení je SysML schopen modelovat širokou škálu systémů, které mohou zahrnovat hardware, software, informace, procesy, personál a zařízení.[1]

- SysML je poměrně malý jazyk, díky čemuž je možnost se ho snadněji naučit a následně používat. Jelikož SysML odstraňuje mnoho konstrukcí zaměřených na software UML, celkový modelovací jazyk je menší jak v typech diagramů, tak v celkových konstrukcích.[20]

- SysML obsahuje alokační tabulky, které podporují i běžné druhy alokací. Zatímco UML poskytuje pouze omezenou podporu pro tabulkové zápisy, SysML v tomto ohledu flexibilnější. Existuje zde funkcionalita pro alokaci požadavků, funkční alokaci a strukturální alokaci. Tato schopnost usnadňuje automatizované ověřování a validaci (V and V) a analýzu mezer (gaps).[20]

- Správa modelů SysML vytváří podpůrné modely, hlediska a pohledy. Tyto konstrukce rozšiřují možnosti UML a jsou architektonicky sladěny s IEEE-Std-1471-2000 (doporučená praxe IEEE pro architektonický popis softwarově náročných systémů).[20]

SysML má k dispozici sedm ze čtrnácti diagramů používaných v UML 2 a přidává "své"dva diagramy (diagramy požadavků a parametrické diagramy), čímž dává k dispozici celkem devět typů diagramů. Mezi nové nebo pozměněné diagramy patří tyto následující:

- Diagram požadavků: Toto je statický strukturální diagram, který ukazuje systémové požadavky a jejich vztahy k ostatním prvkům v modelu. Diagramy požadavků se používají ke specifikaci funkčních (funkce, kterou musí systém vykonávat) a nefunkčních (kritéria používaná k testování účinnosti funkce) požadavků v rámci modelu.[20]

- Parametrický diagram: Parametrický diagram se používá k vynucení matematických pravidel a omezení. Parametrický diagram může například definovat výkonová a kvantitativní omezení, jako je maximální počet připojení, celkový počet transakcí a tak dále. Jedná se o podmnožinu Vnitřního blokového schématu.[20]

- Diagram definice bloku: Blok je komponenta systému, kterou může být ku příkladu software, hardware, organizace, zařízení a mnohé další. Diagram definice bloku je používaný k vizualizaci systémových komponent a jejich obsahu, rozhraní a vztahů. Tento typ diagramu se používá k určení statických struktur, které se používají pro řídicí objekty, datové objekty a objekty rozhraní.[1]

- Vnitřní blokové schéma: Toto je statické strukturální schéma, které představuje konkrétní blok. Diagram ukazuje strukturální obsah bloku (části, vlastnosti, konektory, porty a rozhraní).[1]

Pro představu porovnání využití UML a SysML si lze představit například modelování leteckého systému. Se SysML lze použít diagramy požadavků k efektivnímu zachycení požadavků na funkčnost, výkon a rozhraní, zatímco využití UML podléhá omezení diagramů případů užití pro definování funkčních požadavků na vysoké úrovni. Obdobně lze pomocí SysML použít parametrické diagramy k přesné definici výkonu a kvantitativních omezení, jako je maximální rychlost, maximální vzletová či pohotovostní hmotnost a celková kapacita míst v letadle. Dá se říci, že UML neposkytuje žádný přímý mechanismus pro zachycení tohoto druhu základních informací o výkonu a kvantitativních informacích.

Pokud jde o zbytek leteckého systému, vylepšené diagramy aktivit a diagramy stavových strojů lze použít ke specifikaci integrované softwarové řídicí logiky a informačních toků pro palubní počítače. Jiné strukturální diagramy a diagramy chování v SysML lze použít k modelování továren, které vyrábějí letadla, stejně tak jako rozhraní mezi letištní věží, letadly ve vzduchu nebo piloty připravenými k letu.

4 Analýza diagramů chování

Jak již bylo řečeno v jedné z předchozích kapitol, diagramy chování představují dynamické aspekty systémů. Popisují, co se má v modelovaném systému dít a jaké procesy v něm probíhají. Díky tomu, že diagramy chování ilustrují chování systému, jsou běžně používány k popisu funkčnosti softwarových systémů. Ve zkratce lze říci, že vizualizují, specifikují, konstruují a dokumentují dynamické aspekty systému. V této kapitole bude popsány tři nejvýznamnější diagramy této kategorie.

4.1 Use Case diagramy

Lze jednoduše říci, že "diagramy případů užití jsou základními plány navrhovaného systému". [22]

Diagram případu užití zastává grafické znázornění veškerých existujících interakcí mezi uživatelem (v diagramu nazvaným jako hercem) a daným systémem. Modelovaný systém je znázorněn pomocí několika jednotlivých případů užití, a ty jsou s herci spojeny pomocí vazeb. Herci mohou být osoby, zařízení, organizace nebo softwarové a hardwarové součásti systému. Herců v diagramu může být několik a každý z nich může interagovat s libovolným množstvím případů užití. Případy užití bývají prezentovány buďto kruhy nebo elipsami (vytvářející určité bubliny). Herci/uživatelé jsou vyobrazeni jako panáčci. Jednotlivé případy užití mohou být navázány na další "bubliny"se systémvými akcemi pomocí dvou různých spojení. Prvním je "extend", který dává možnost přídavné funkcionality pro danou interakci, která však není povinná. Lze si pod tím představit například poplatek za použití bankomatu jiné banky - tedy poplatek není povinný pro majitele karty stejné banky, čímž pádem nastane tato akce pouze v některých případech. Oproti tomu je include povinná akce u některé z interakcí a zpravidla bývá sdílena/znovupoužita více funkcionalitami (bublinami), jež jsou bez ní nekompletní. Pokud by se include nacházelo ve spojení s pouze jednou další bublinou akce, postrádalo by v tu chvíli smysl a mělo by být zakomponováno do scénáře dané interakce. Jako příklad může posloužit opět bankomat - ale v tomto případě kontrola PINu karty, kdy je potřeba kontrolovat při každé akci (vybrání částky, uložení peněz, kontrola stavu účtu) zda je správně zadaný vůči vložené kartě. [23]

Samotný případ užití zpravidla vždy proniká do různých podrobných možností dané interakce, které se v něm mohou stát. Je to tedy posloupnost určitých akcí, které se v sys-

tému provádí, aby dodaly hercům/uživatelům určité viditelné a přínosné výsledky. Každá interakce obsahuje scénář, ve kterém je popsáno vše, co je nutné aby se odehrálo, aby bylo možné dojít k danému výsledku. Mohou v nich existovat také alternativní cesty, nebo možnosti využití jiných use casů, které jsou připojeny pomocí "extend"vazby. Use case diagram však jako celek poskytuje obecný pohled na systém na vyšší úrovni (tzv. High level pohled). Díky faktu, že jsou use case diagramy poměrně jednoduché a přehledné, jsou dobrým komunikačním prostředkem mezi všemi zúčastněnými stranami návrhu systému, tedy od analytiků, přes vývojáře až po manažerské pozice. Cílem diagramů je ukázat těmto stranám, jak daný model napodobuje chování ve skutečném světě a jakým způsobem bude i mimo jiné na základě toho navržen. V rámci use case diagramů byly na toto téma provedeny různé výzkumy za účelem zjištění, zda jsou ještě v dnešní době nutné (a to především pro chápání výše postavených představitelů firem). Bylo docíleno toho výsledku, že tyto diagramy bez jakýchkoliv problémů ukázaly jasný záměr toho, co má systém konat a byly pochopeny a interpretovány kompletněji, oproti jiným diagramům a způsobům znázornění systému. [23]

Use case diagramy by vždy měly zachycovat funkční požadavky systému. Z toho důvodu by v modelovaném systému měly funkční požadavky převládat nad těmi nefunkčními, dále je vhodné, aby měl systém ideálně více typů uživatelů, kteří mají přístup k různým funckionalitám, nicméně i jeden uživatel je pro modelování akceptovatelný. [23]

4.2 Diagramy aktivit

Diagramy aktivit reprezentují grafický zápis pracovních postupů navazujících činností a akcí s možností výběru, opakování a jejich koordinace. Jednotlivé činnosti mohou být koordinovány za účelem poskytování rozmanitých služeb, které se mohou nacházet na různých úrovních abstrakce vůči jejich reálnému obrazu. Koordinace se využívá zejména v místech, kde se v use casech jednotlivé činnosti navzájem kryjí. Vynechání koordinace v takových chvílích nepovede ke kýženým výsledkům v daném toku. V UML je diagram aktivit vytvořen pro modelování jak výpočetních a organizačních procesů (tedy pracovních toků), tak i datových toků, které se protínají se souvisejícími aktivitami. Ačkoliv je primárním cílem diagramů aktivit zobrazení celkového toku řízení, mohou v nich být také obsaženy určité prvky, které znázorňují již zmíněný tok dat mezi jednotlivými aktivitami, a to pomocí jednoho nebo více datových úložišť. Aktivita může být také vytvořena k libovolnému prvku, jenž určuje její kontext za účelem modelování jeho chování. Může být tedy přidělena

kromě případů užití také k třídám, rozhraním, komponentám, kolaboracím/komunikacím či k jednoduchým operacím. Z modelovacího hlediska je to síť uzlů, které jsou spojovány pomocí tzv. hran, znázorněných jako šipky. Uzel může být provedení podřízeného chování, jako jsou aritmetické výpočty, volání operací nebo manipulace s obsahem objektu. Činnosti mohou tvořit hierarchie vyvolávání, které uvádějí v pohyb další činnosti, a ty se nakonec rozhodují pro jednotlivé akce. V objektově orientovaném modelu jsou aktivity obvykle vyvolávány nepřímo jako metody vázané na operace. Uzlů existují 3 kategorie. Akční uzly představují samostatné jednotky určité práce, které jsou v rámci aktivity dále nedělitelné. Kontrolní uzly určují tok skrze aktivitu a objektové uzly reprezentují objekty v aktivitě použité. Všechny aktivity začínají jedním kontrolním uzlem (počátečním) a mají také jeden či více konečných uzlů (může existovat více výsledků dané interakce). Průchod diagramem lze popsat jako tokenovou hru. Tato hra ukazuje tok tokenů skrze síť uzlů propojených hranami za působení určitých pravidel. Jako tokeny lze v diagramu označit tok kontroly, objekt nebo nějaká využívaná data. V jakékoliv chvíli lze určit stav diagramu podle rozmístění jeho tokenů. Tokeny se pohybují ze zdrojového uzlu do cílového po již zmíněných hranách. Pohyb tokenu je závislý na určitých podmínkách a může k němu dojít pouze v tom případě, že jsou všechny splněny. Formát podmínek se může lišit, záleží na typu použitého uzlu. U kontrolních uzlů je určováno, jakým způsobem jsou tokeny posílány ze vstupních hran na výstupní. Pro příklad, počáteční uzel zahajuje aktivitu, finální ji ukončuje a mezi tím nacházející se spojovací uzel může nabídnout token na své jedné výstupní hraně pouze tehdy, pokud byly tokeny zaslány na všechny jeho vstupní hrany. Tím dochází ke kontrole správného a kompletního vstupu do uzlu a teprve poté je možné pokračovat v průchodu diagramem. [24]

Aby byly diagramy aktivit více přehledné a lépe čitelné, lze použít jejich rozdělení jak z vertikálního, horizontálního hlediska a dokonce také pomocí zakřivených linií. Každé rozdělení aktivit představuje logické seskupení spolu souvisejících akcí z vyššího pohledu na modelovaný problém. I zde, jako už bylo zmíněno v práci dříve, se rozdělení nazývají swimlanes, tedy plavecké dráhy. Jejich přidání do diagramu může mít na něj velký vliv, jelikož je následně vše snazší na pochopení. Dalo by se říci, že pro dělení diagramů aktivit neexistují v podstatě žádná pravidla, lze je tedy rozdělovat "dle libosti". Mnohokrát jsou využívány pro vyznačení jednotlivých use casů, tříd, kompoment, rolí či organizačních jednotek (přede-

vším v business modelování). Každá sada oddílů v rozdělení by měla mít jednu nadřazenou dimenzi, pomocí které bude popsána základní sémantika seskupených prvků v ní. V rámci této dimenze mohou být jednotlivé oddíly hierarchicky vnořovány.[24]

Lze si představit mezinárodní firmu, která má rozdělena jednotlivá odvětví do míst v různých zemích. Nadřazená dimenze může mít název "Město", oddíly v ní mohou být například Praha a Berlín, přičemž Praha je rozdělena na 3 oddělení. Analýza, vývoj a nasazení tedy probíhají v Praze, zatímco testování se koná pouze v Berlíně. Token započne svou cestu na počátečním uzlu v Praze a poté je po analýze a finalizaci vývoje zaslán na testování ve vedlejší swimlane (Berlín). Po úspěšném ukončení testování je přesunut na další činnost v aktivitě, zpět do Prahy, tedy nasazení, a poté končí svou cestu v konečném uzlu. Výsledný diagram je jasnější a jednodušeji pochopitelný, než kdyby dělení neexistovala a bylo vše pouze na jednom místě. Nejčastěji jsou diagramy aktivit používány pro analýzy pracovních postupů, v designu a již zmíněném modelování business procesů. V rámci pracovních postupů bývají buďto použity pro zobrazení toků v daném use casu pro snazší pochopení u všech zúčastněných stran nebo pro modelování toků mezi jednotlivými use casy. To je znázorněno ve speciálním typu diagramu aktivit zvaného přehledový diagram. V designu se používají pro modelování detailu operací či algoritmů. [24]

Podstatou správně navrženého diagramu aktivit je, že je zaměřen na chování jednoho specifického aspektu dynamického chování systému. Kvůli tomu je nutná zvolit již zmíněnou správnou míru abstrakce, aby bylo možné vynechat nepodstatné elementy. Tím bude zajištěno, že komunikace pomocí modelu přenese požadovanou zprávu k cílové skupině uživatelů. Není potřeba dáv to diagramu veškeré dostupné údaje a informace, ale pouze takové množství, aby zůstala zachována chtěná pointa. Vytvořit "objemný"diagram, ze kterého nebude odstraněno nic z veškerých dostupných informací není příliš složité, ale právě složitost přichází s tím, aby byl tvůrce diagramu shcopen rozeznat, co je ještě prospěšné pro modelování, a co už by v diagramu překáželo a bylo graficky zobrazeno zbytečně. Z toho důvodu bývá nejlepší držet se rčení, že v jednoduchosti je síla.[24]

4.3 Stavové diagramy

State machine diagrams (tedy diagramy stavových strojů) mají za úkol modelovat dynamické prvky chování systému, podobně jako aktivity diagramy. Existuje zde však řada

rozdílů, a tou hlavní je jiný účel jejich modelování a také rozdílná sémantika, již používají. Zatímco diagramy aktivit stojí na základech jsou běžně využívány zejména pro vytváření modelů obchodních procesů (a tím pádem pracují s různým počtem objektů), stavové stroje modelují historii životního cyklu pouze jednoto objektu. Ten je ve výsledku zobrazen jako stroj, který má jeden nebo více konečných stavů. U takovéhoto stroje dochází k přechodům mezi stavy pomocí přesně specifikovaných způsobů, které jsou reakcí na proběhnuvší události. [25]

Jakožto tři hlavní prvky stavových strojů lze označit stav, událost a přechod. Stavem se rozumí určitá situace v průběhu existence objektu. V jejím rámci může vykonávat určité činnosti, splňovat podmínky, které by znamenaly přechod dál nebo může vyčkávat na vypuknutí nějaké události. V průběhu času to znamená, že se jeho stav postupně mění. Zároveň lze však kdykoliv jeho stav přesně určit, a to díky hodnotám jeho atributů, dle vztahů vůči dalším objektům či dle aktivit, jež právě provádí. V rámci jejich existence, si objekty mezi sebou posílají zprávy, a tyto zprávy nejsou nic jiného než určité události, které mohou způsobit změnu stavu jednoho z objektů. Pro jednoduchý příklad je možné si vzít žárovku. Z pohledu jejího používání nám nejde o to, jaký jde do ní proud, nebo jaká jí zbývá životnost, důležité je pouze to, zda je zapnutá, vypnutá a nebo jestli je ve svém finálním stavu, tedy psasklá/vyhořelá. Vždy je potřeba najít pouze ty stavy, které konají v systému nějaké viditelné a podstatné změny. Jedině ty je vhodné modelovat ve stavovém diagramu. [25]

Stavové stroje bývají nejčastěji používány pro modelování chování v use casech nebo třídách. Každá třída může jen jeden stavový stroj chování, ve němž se modelují všechny jeho stavy, události, přechody a instance dané třídy. Každá třída může mít navíc jeden či více stavových strojů protokolu. Ty jsou ovšem převážně používány pro vytváření modelů klasifikátorů, které nemají žádné chování. Pokud má třída rodičovskou třídu, dědí stavový stroj protokolu právě od ní. Pokud nastane situace, že by nějaká třída měla více než jeden stavový stroj, je nutné, aby byly vůči sobě navzájem konzistentní. [25]

Behavior state machine používá své stavy, přechody a události, aby popsal kontextové klasifikátory. Pokud však neexistuje u daného klasifikátoru žádné chování pro modelování, nelze ho z tohoto důvodu použít. Příkladem takových klasifikátorů mohou být zejména porty a rozhraní. Ty se používají pouze pro definici protokolů.[25]

Podobně jako stroje chování, mají také stavové stroje protokolu definované stavy, přechody a události. Zde jsou však využívány pro popis protokolu kontextových klasifikátorů. V protokolu jsou vypsány podmínky, díky kterým se následně volají určité operace, a to buď na klasifikátoru samotném, nebo na některé z jeho instancí. Mimo jiné obsahuje výsledky volání operací a také řazení volání operací. Stavové stroje protokolu nepopisují implementaci chování, pouze ukazují, jakým způsobem se chování jeví nějaké vnější entitě. Tyto stavové stroje mohou být použity pro definici protokolu pro všechny klasifikátory, ať už mají nějakou implementaci, nebo ne. Stavy v těchto strojích nemohou upřesňovat akce, to je možné pouze u strojů chování. Při modelování je lze označit pomocí klíčového slova protocol za názvem daného stavového stroje.[25]

Pro uvedení příkladu diagramu stavových strojů lze víz v potaz jednoduchý příklad z reálného světa, který neustále mění své stavy, a tím pádem opakovaně prochází statovým strojem, je žárovka. Zjednodušeně lze říci, že se jedná o zasílání událostí žárovce pomocé přepínání tlačítka. Dvě události, které se zasílají, jsou zapnout (v modelu značeno jako přísun proudu do žárovky) a vypnout (přerušení proudu). Diagram stavového stroje obsahuje právě jeden stavový stroj pro jeden reaktivní objekt. V tomto případě sestává tento objekt ze žárovky, přepínače (tlačítka) a elektrického zdroje (proudu). Obsahem diagramu jsou stavy, vyobrazené jako obdélníky se zaoblenými hranami, počáteční stav je modelován jako vyplněný kruh, zatímco koncový stav jako vyplněný kruh s kruhem okolo něj. Přechody ukazují všechny možné cesty mezi jednotlivými stavy a onzačují se pomocí šipek. Události jsou psané u přechodů, které mají spouštět. Zpět ke příkladu se žárovkou, když se přepne spínač, událost zapnout je zaslána k žárovce. Události jsou okamžité, což znamená, že mezi zasláním události pomocí spínače a jejím doražením k žárovce neuplyne žádný čas. To umožňuje ve teorii jejich lepší sledovatelnost. V opačném případě by mohlo dojít k možnosti, že by 2 a více události závodily, která se dostane první do dalšího stavu a bylo by nutné modelovat podmínky takového "závodu"do modelu.[25]

Co se týče syntaxe stavu v diagramu, je vždy povinné jeho jméno. Poté může každý stav obsahovat nula a více akcí či aktivit. Akce by vždy měly být okamžité a nepřerušitelné, zatímco aktivity by měly trvat dané a konečné množství času a jsou přerušitelné. Každá akce ve stavu je spojena s jeho vnitřním přechodem, jenž je vyvolán na základě nějaké události. V každém stavu může být libovolné množství nejen akcí, jak již bylo zmíněno, ale

také vnitřních přechodů. Vnitřní přechod značí, že se událo něco, co stojí za to, aby bylo znázorněno v modelu, avšak ne tak razantního, aby to zapříčinilo přechod do nového stavu. Při přihlašování by takový přechod mohl být například zmáčknutí tlačítka pro zobrazení pomoci, či nápovědy. Ve stavech se také nacházejí 2 speciální typy akcí, vstupní a výstupní, které jsou asociovány se stejně zvanými událostmi. Vstupní událost nastává okamžitě a automaticky, a je to první věc, která se stane, když je přestoupeno do daného stavu a zároveň vykoná přidruženou akci. Stejně probíhá i výstupní událost, kde se událost i akce stanou při odchodu z daného stavu. [25]

Přechody v diagramech stavových strojů v předchozím odstavci zmíněné vnitřní stavy (tedy vnořené), nebo vnější stavy, které jsou modelovány ve formě šipek. Jakýkoliv přechod může mít 3 volitelné prvky. Prvním je nula nebo více událostí, které specifikují vnější nebo vnitřní aféry, jež mohou započít přechod. Dále je to nula nebo jedna hlídací podmínka, což je v tomto případě Booleanovská hodnota, která musí být nastavena na hodnotu true, než přechod proběhnout. nastává vždy po události.Posledním prvkem/prvky, které může přechod obsahovat je nula a více akcí. Jedná se o činnosti spojené s přechodem a nastávájí, když k přechodu dojde. Přechody mohou být spolu propojeny pomocí tzv. pseudo stavů. Ty představují body v diagramu, kde dochází buďto ke sjednocování, nebo rozdělování přechodů (tedy šipek). "Křižovatka"pseudo stavu může mít více než jeden přechod. Pokud je tomu tak, poté každý výstupní přechod musí být chráněn vzájemně se vylučující ochrannou podmínkou, aby správně došlo k vybrání pouze jednoho z nich. Existuje také výběrový pseudo stav, který umožňuje řídit průchod diagramem pomocí ručně definovaných podmínek u každého výstupního přechodu. Platí zde, stejně jako v předchozím případě, že musí být chráněny vzájemně se vylučujícími podmínkami.[25]

Události označují specifikace určitých důležitých situací a jsou pomocí nich spouštěny přechody. Lze popsat čtyři typy událostí:

- Událost volání - Je požadavkem pro konkrétní operaci, jež by měla být volána na instanci kontextové třídy. Je nutné, aby volání mělo stejnou identifikaci, kterou má tato třída. Operace je provedena na základě příjmu události volání. Každá událost volání může mít specifikovanou sekvenci akcí, které jsou od sebe běžně oddělovány pomocí středníků. Akce z těchto sekvencí udávají sémantiku dané operace, navíc mo-

hou využít atributy či operace z její kontextové třídy. Operace s návratovým typem ho musí také mít v událostech volání. [26]

- Signální událost - Signál se dá označit jako balíček určitých informací. Ten je po jejich „zabalení“ asynchronně zasílán mezi objekty. Ve stavovém stroji je modelován jako třída, jež má ve svých atributech uchované informace za účelem jejich sdělování dál. Signál zpravidla nemá žádné operace, jelikož jeho jedinou rolí je přenos informací. Zaslaný signál je zobrazen pomocí pětiúhelníku ve tvaru šipky, jenž má uvnitř svůj název. Přijetí signálu se značí jako vydutý pětiúhelník.[26]

- Změnová událost - je specifikována jako booleovský výraz. Akce spojené s touto událostí jsou provedeny právě tehdy, kdy se hodnota v booleanu zmení z false na true. Všechny hodnoty v booleovských výrazech musí být konstanty, globální hodnoty nebo atributy či operace kontextové třídy. Z hlediska implementace změnová událost znamená opakované testování booleovské podmínky v daném stavu. Aby mohla být taková událost znovu spuštěna, musí se dostat zpět do stavu false a následně opět do true. [26]

- Časová událost - tuto kategorií událostí lze nejčastěji poznat pomocí klíčových slov when a after. Klíčové slovo when specifikuje konkrétní čas, ve kterém je daná událost spuštněna a slovo after určuje prahovou dobu, po které se spustí událost navazující. Lze zapsat například when(date = 20/10/2022) a after(2 days). Je správné se ujistit, že jsou jednotky času zaznamenávány do diagramu pro každou časovou událost. Stejně jako u předchozího typu události, i zde všechny hodnoty a symboly ve výrazech musí být konstanty, globální hodnoty nebo atributy či operace dané kontextové třídy. [26]

5 Využití diagramů chování

V této kapitole budou představeny diagramy chování navržené v modelovacích jazycích či notacích již zmíněných v předchozích kapitolách, tedy v UML 2.0, BPMN 2.0 a SysML. Motivem, který budou tyto diagramy pokrývat, je téma udržitelnosti. Přesněji řečeno tedy její část, která se soustředí na elektromobilitu. V následujících podkapitole bude udržitelnost a elektromobilita blíže rozebrána a následně na ni budou navazovat již samotné vytvořené diagramy k elektromobilitě se vztahující.

5.1 Udržitelnost a elektromobilita

Než se přejde k popisu vytvořených diagramů, bude v této části nastíněno téma, okolo kterého budou diagramy modelovány. Jak již bylo předem představeno, je jím elektromobilita.

Elektromobilitu lze definovat jako silniční dopravní systém založený na vozidlech, která jsou poháněna elektřinou. Některá z těchto vozidel mohou být vybavena technologiemi, jež jim dovoluje vyrábět vlastní elektrickou energii (tedy hybridní vozy). Další typy vozidel používají elektrickou energii ze zdroje, který se nachází mimo vozidlo samotné, a to nejběžněji z elektrické sítě. To funguje bez problémů nejen pro elektromobily, ale také pro vozidla, která elektrickou energii neuchovávají. Těmi jsou například trolejbusy, nicméně i ty mnohdy mívají záložní baterie pro případ výpadku elektřiny. Dopravní systém, který využívá elektřinu ze sítě, může jednoduše čerpat energii z velkého výběru zdrojů, a to bez větších úprav elektrických vozidel či systémů dodávávání energií. Podstatným faktorem elektromobility je fakt, že může (a to především ve městech) také přispět ke snížení emisí CO2. To platí zejména v případě, že je elektřina vyráběna z obnovitelných zdrojů. Pokud však elektromobily zůstanou u využívání elektrické energie vyráběné ku příkladu z uhlí, výsledné klimatické dopady elektro pohonu by mohly být poněkud negativní, a to i v porovnání s vozidly, která jsou poháněna fosilními palivy. To je v tuto chvíli jeden z hlavních problémů, díky kterým nelze na elektromobily pohlížet jako na "čistou"variantu cestování. Technologický vývoj je mnohdy řízen a veden určitým směrem díky preferencím uživatelů, ale také možností různých firem získat konkurenční výhody (a tím pádem také často vyšší zisky) oproti ostatním společnostem v daném oboru. V případě rozvoje elektromobility však

uživatelé nehrají až tak velkou roli. Obrovská část řidičů je s vozy se spalovacími motory nadmíru spokojena. Pokud se navíc vezme v potaz cena elektromobilů, výdrž baterie a cena její případné opravy či výměny, je pro zákazníky většina řešení elektrických vozidel podstatně horší, než u automobilů se spalovacími motory. Navíc nejsou fosilní paliva natolik drahá, aby touto cestou přišla uživatelům větší motivace pro nákup dražších elektromobilů. Obecně řečeno lze konstatovat, že tržní síla elektromobilů není taková, aby by to pohlo se zákaznickou poptávkou, aniž by přišla nějaká významnější politická opatření. Jak již bylo napsáno, mnohé společnosti prosazují elektromobilitu, aby byly napřed oproti ostatním, a to i přes mnohdy nedostatečnou poptávku. [27]

Dalším důvodem, proč mohou lidé váhat s přechodem na elektrické vozy, může být prozatimní nedostatek nabíjecích stanic na určitých místech. V oblasti elektrifikaci dopravní obsluhy se vyskytlo několik různých trendů. Jedním z nich může být elektrifikace silnic, coby platná varianta vůči klasickým silnicím. Hnacím faktorem pro komplexní elektrifikaci by mohl být přechod k elektromobilům, jež mají menší baterie, které se však rychleji a zároveň snadněji nabíjejí. V rámci takovýchto konceptů je nutné vnímat také dopravní spoje. V těch má totiž elektromobilita pro dosažení cílů snížení emisí téměř hlavní pozici. Příkladem toho jsou již dlouhou dobu elektrické vlaky, které pomáhají snížit nejen náklady na provoz, ale také celkové emise, jež produkuje nákladní a osobní doprava. Propojení rozmanitých typů dopravy s cílem redukce emisí vytváří předpoklady pro uvedení konceptů tzv. mobileenergy-as-a-service, jenž má propojovat infrastrukturu měst, energetiku a především právě dopravu do jednoho velkého celku. Cílem tohoto konceptu je přijít se systémem, který bude schopen podpory udržitelné dopravy ve velkých městech a metropolích. Pro to, aby byl úspěšně dosažen přechod z aut se spalovacími motory na vozidla používající elektromotory, je klíčový robustní systém pro poskytování služeb nabíjecích stanic. Pro plánování sítí dobíjecích stanic již v dnešní době existují metodiky, které se používají pro výzkum toho, zda je taková síť schopna ve velkých městech schopna současné napájet velké množství vozů s elektrickým pohonem. To samé platí u plánování pro jejich vhodné umístění. V rámci budování sítě těchto stanic je potřeba brát v potaz úvahu, že je nutné, aby síť byla strategicky postavena tak, že bude možné s těmito vozidly cestovat i do mimo městských oblastí. Aby vzrostla poptávka po (především osobních) elektromobilech, je nutné postavit větší množství stanic pro čerpání elektrické energie. Pro začátek je nutné, aby nabíjecí stanice byly k dispozici na

parkovištích. Během toho, co lidé pracují, jsou doma, nebo v nákupních centrech, mohou se jejich elektromobily nabíjet a díky tomu rostou jejich pozitivní spotřebitelské preference vůči elektromobilitě jako takové. Existuje několik podob doporučených nabíjecích technologií, které přispívají k vytváření a realizaci konceptu chytrého města. Jsou to grid-to-vehicle, vehicle-to-grid (to je v situacích, kdy vozidlo může díky solárním paneům získávat energii, kterou následně vrací do sítě), parking lot-to-vehicle (zde parkoviště představuje prvek systému pro dobíjení vozidel) a parking lot-to-grid. V rámci dosažení výsledků v udržitelnosti je vhodné spojení dobíjecích stanic s fotovoltaickými články. Zde by byl vhodným a efektivním způsobem ovládání nabíjecích stanic decentralizované řízení nabíjení. [28]

V posledních letech je již trh s elektromobily poměrně rozšířený, oproti jejich počátkům zhruba dekádu zpět. Celosvětově bylo v roce 2020 k dispozici asi 370 modelů elektromobilů, což je o zrhuba 40 % více, než v roce 2019. Čína má nejširší nabídku, což odráží její méně konsolidovaný automobilový sektor, a to, že jde o největší světový trh nejen s elektromobily .V roce 2020 byl však největší nárůst počtu modelů v Evropě, kde se více než zdvojnásobil. Modely bateriových elektromobilů jsou nabízeny ve většině segmentů vozidel skrze všechny regiony, plug-in hybridní vozy jsou vychýleny spíše směrem k segmentu větších vozidel. Modely SUV představují zrhuba polovinu všech dostupných modelů elektromobilů na trzích na celém světě. Modely, které jsou k dispozici v Číně, tvoří dvojnásobek počtu modelů dostupných v Evropě a dokonce čtyřnásobek modelů dostupných ve Spojených státech amerických. Takovýto rozdíl je částečně vysvětlitelný podstatně nižší vyspělostí trhu s elektromobily v USA. To také odráží slabší regulace a pobídky ke koupi elektromobilů na celonárodní úrovni. Průměrný jízdní dosah nových bateriových elektromobilů se neustále zvyšuje. V roce 2020 byl vážený průměrný dojezd pro nový bateriový elektromobil asi 350 kilometrů (km), oproti 200 km v roce 2015. V roce 2022 již existují modely elektromobilů s dojezdem mezi 750 až 850 kilometry. U plug-in hybridů zůstal průměrný elektrický dojezd v posledních několika letech poměrně konstantní, a to kolem 50 až 60 km. [29]

5.2 UML 2.0 diagramy

5.2.1 Use Case diagram

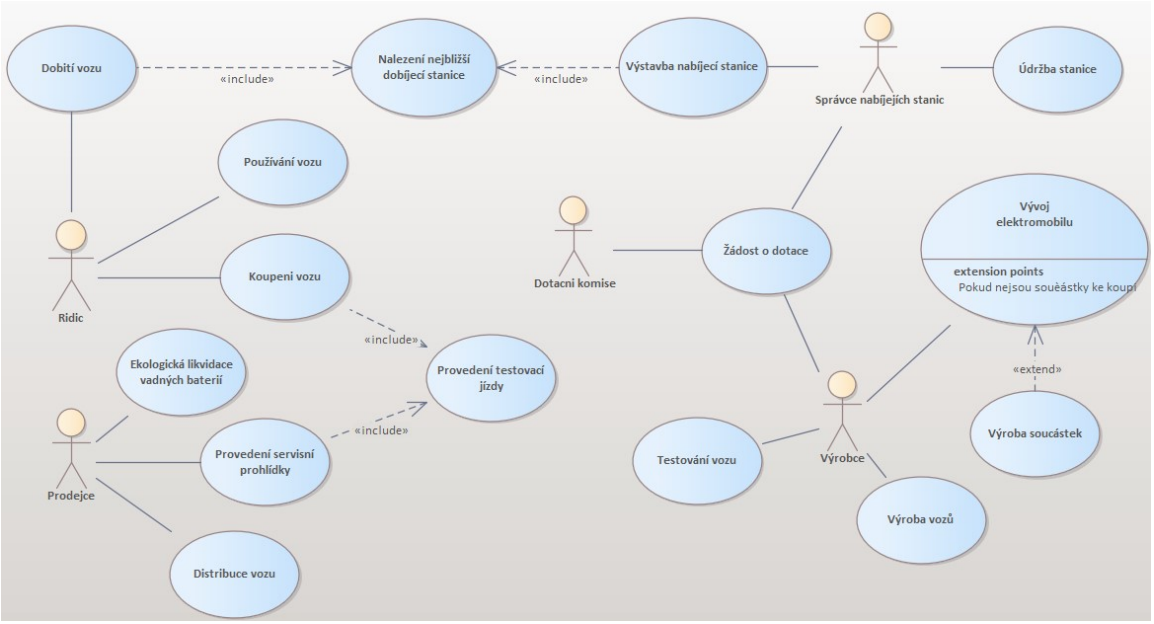
Základním kamenem pro samotný začátek modelování libovolného systému nebo procesu bude určení, co se má v modelované situaci dít, jaké role buduo v systému či procesu figurovat a samozřejmě jaké mezi nimi mají být vazby.

Na obrázku 2 lze vidět jednotlivé role a jejich případy užití. V rámci tématu elektromobility bylo vybráno pět rolí, kterými jsou řidič, prodejce elektromobilů, správce nabíjecích stanic, dotační komise a výrobce elektromobilů. Již u rolí lze vidět určitou míru abstrakce, jelikož právě výrobce elektromobilů by mohl být dále rozdělen například na management, který rozhoduje o vývoji nových modelů, na designéry, vývojáře pohonných jednotek, osoby určené pro testování prototypů a mnoho dalších.

Dalo by se říci, že nejpodstatnějším aktérem v tomto diagramu je výrobce elektromobilů. Bez jeho existence by nebylo možné provádět téměř žádné další případy užití, které vykonávají ostatní role. Mezi jeho případy užití výrobce patří výrobu vozů, testování vozů, možnost zažádání o dotace, a také vývoj nových modelů elektromobilů, na který je přes vazbu extend napojen use case výroba součástek. Ta je zde pod podmínkou, že dané součástky nutné pro vývoj a výrobu prototypu nového modelu nebudou k dispozici a ani je nebude možné zakoupit. Poslední use case výrobce je možnost žádosti o dotaci, kterou má jako svůj případ užití také uživatel dotační komise pro schvalování dotací. Žádost o dotace je také sdílena se správcem nabíjecích stanic, u kterého může být použita na spolufinancování výstavby nových a údržbu stávajících stanic. Tyto dva zmíněné případy jsou také dalšími use case pro roli správce. Výstavba nové nabíjecí stanice má k sobě přes vazbu include připojen use case nalezení nejbližší dobíjecí stanice, a to z důvodu, aby nebyly postaveny přílíš blízko, nebo naopak nebyly příliš daleko od sebe. Aktér řidič má dva use casy. Tím prvním je dobití vozu, které je vazbou include také připojeno k případu nalezení nejbližší stanice, aby bylo možné včas dorazit s elektromobilem ke stanici v situaci, kdy začne hlásit nízký dojezd. Řidič bude mít samozřejmě také use case pro používání vozidla a use case pro zakoupení vozidla. Ke druhému jmenovanému je pomocí vazby include připojeno provedení testovací jízdy. Posledním aktérem je prodejce, který má na starosti distribuci nových vozů, ekoloigckou lidkvidaci starých a vadných baterií a nakonec zařizuje servisní prohlídky,

jež mají include vazbu na provedení testovací jízdy, z důvodu zjištění případných problémů.

Diagram případů užití tímto způsobem poskytuje sice základní, ale zároveň high level pohled na modelované business procesy.



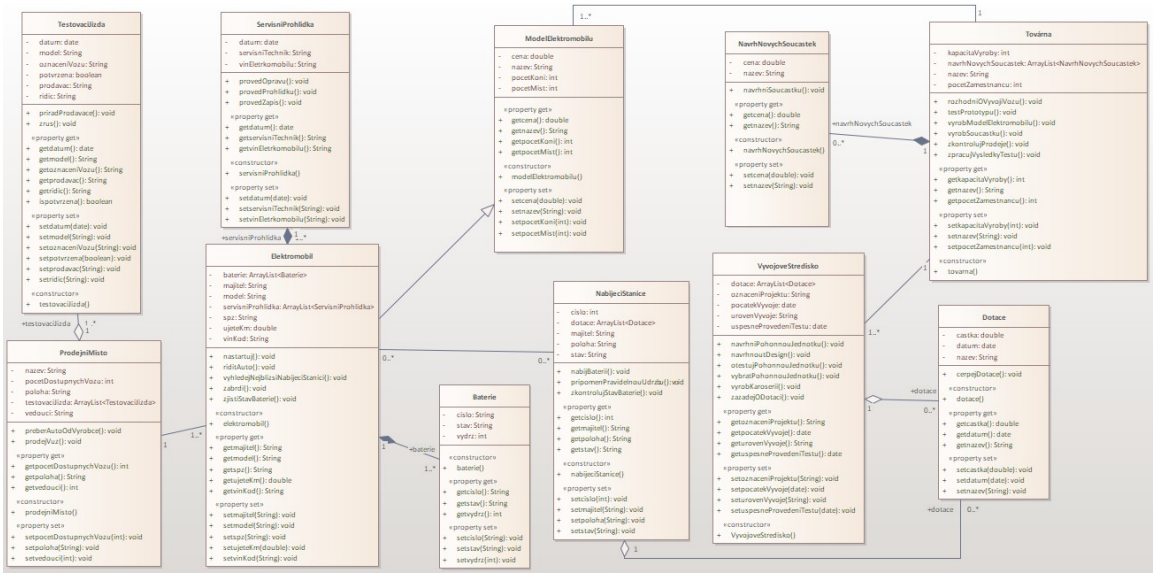
Obrázek 2: Use case diagram, Zdroj: Vlastní zpracování

5.2.2 Class diagram

Přestože se nejedná o diagram chování, bylo by dobré alespoň obecně popsat také diagram tříd. Ten je možné vidět na obrázku 3 a je v projektu vytvořen spíše pro potřeby uvědomění si, jak by spolu mohly jednotlivé entity komunikovat a spolupracovat. Nicméně bylo by možné jej použít pro budoucí potřeby při vytváření databáze systému, či základnímu návrhu tříd v přidružené aplikaci. Dalo by se říci, že v tuto chvíli se v diagramu nacházejí základní stavební prvky potřebné pro realizaci takového systému, který by bylo následně dle potřeby možné nadále rozšiřovat.

S popisem tříd lze začít u továrny, ve které probíhají veškerá rozhodnutí o případném vývoji nových vozů, jejich testování a zpracování výsledků testů. Je na ní také závislý návrh či výroba součástek a vybírají se zde modely elektromobilů pro výrobu. Na továrnu je navázána třída NavrhNovychSoucastek, ModelElektromobilu a VyvojoveStredisko, se kterou je

připojena třída Dotace. Již zmíněný ModelElektromobilu je rodičovská třída obsahující základní údaje o daném modelu vozu, jejíž dceřinnou třídou je Elektromobil. Ten představuje jednotlivé vyrobené vozy. Na ni jsou posléze navázány třídy ServisniProhlidka, NabijeciStanice (jež má, podobně jako vývoj, možnost čerpání dotací) a Baterie. Poslední třídou propojenou s elektromobilem je ProdejniMisto, které je před jejich koupí také majitelem všech elektromobilů, a díky vazbě se třídou TestovaciJizda mohou pomocí něj vozy zkoušet zákazníci.



Obrázek 3: Class diagram, Zdroj: Vlastní zpracování

5.2.3 Timing diagram

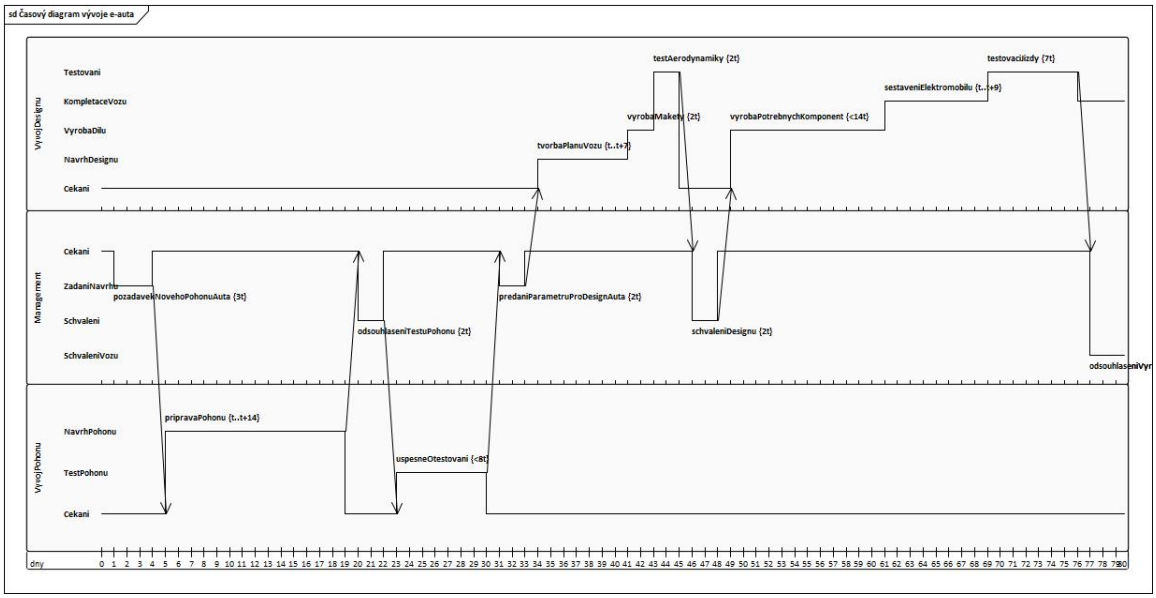
Časový diagram, jak již bylo napsáno, zobrazuje vývoj určité aktivity na lienární časové ose. Jedním z nejvhodnějších kandidátů pro tvorbu tohoto typu diagramu je aktivita vývoje elektromobilu. Maximální doba, po kterou je možné tento automobil vyvíjet, je v tomto případě 80 dní.

Diagram se ponese v duchu "best case scenario", tedy takový průchod diagramem, při kterém nenastanou žádné chyby a vše bude probíhat hladce. V diagramu (obrázek 4) existují tři stavové "životní linie"(state lifelines), kterými jsou v tomto případě tři oddělení v továrně. Těmi jsou management, vývoj pohonu (motorizace) a vývoj designu, do kterého je mimo jiné pro přehlednost diagramu také zahrnuta výroba dílů, následné montování elektromobilu, a také testování.

Každé oddělení má několik stavů pro možné přechody mezi nimi. U managementu je to čekání, zadání návrhu, schválení (slouží pro kontrolu postupu práce na vývoji vozu) a schválení hotového prototypu vozu pro jeho sériovou výrobu. Vývoj pohonu má stav čekání, stav návrhu pohonu a testu pohonu. Design má stejně jako předešlé linie čekání, poté návrh designu, výrobu dílů, kompletaci vozu a jeho testování.

Z pohledu na diagram je jisté, že během vývoje budou vznikat pro jednotlivé účastníky určité prodlevy a bude docházet k čekání (k tomu dochází vždy, když je jedním účastníkem odeslána zpráva na jiného účastníka interakce), což je nejvíce vidět u obou vývojových oddělení. Lze tím pádem počítat s tím, že ve stavech čekání mohou pracovat na jiných projektech, ale v rámci znázornění vývoje není tato informace dostatečně podstatná pro to, aby byla přenesena do modelu.

Celý proces započne v lifeline managementu, kde dojde k první změně stavu z čekání na zadání návrh. V tomto stavu se bude vytvářet požadavek nového pohonu auta, tato fáze bude trvat 3 dny. Na konci této fáze nastane předání zprávy do vývoje pohonu, kde dojde ke změně stavu z čekání na návrh pohonu. Během této fáze dochází k přípravě nového motoru, která může trvat jeden až patnáct dní. To je znázorněno pomocí události se zapsanou dobou trvání pripravaPohonu(t..t+14), kde "t"je jednotka času rovnající se jednomu dni. Po dokončení nastane návrat do managementu, kde dojde k události odsouhlasení pohonu. Následně přejde akce zpět to vývoje pohonu, kde je vymezen maximálně týden (t<8) na úspěšné otestování motoru, zakončeného zasláním zprávy do managementu. Zde má toto oddělení dva dny na vytvoření a předání parametrů pro design a odeslat zprávu na vedlejší oddělení vývoje designu. To přechází z čekání na návrh designu, kde je jeden až osm dní na vytvoření plánů vozu. Poté se ve stavu výroba dílů vytvoří maketa, po jejíž výrobě se přejde do stavu testování a budou zde dva dny na testování v aerodynamickém tunelu. Po dotestování se předává zpráva zpět do managementu a dojde ke schválení designu (opět dva dny). Tato dobrá zpráva je znovu přenesena do vývoje designu, ve které již může dojít k výrobě potřebných součástek a komponent pro daný vůz (t<14, tato fáze musí trvat méně, než 2 týdny). Okamžitě po ní nastává sestavení prototypu elektromobilu (trvající jeden až 10 dní, t..t+9), aby mohly začít týdenní testovací jízdy. Po jejich splnění již nastává poslední zaslání zprávy mezi účastníky na management, ve kterém dojde k poslední události. Tou je konečné odsouhlasení výroby a nachází se na stavu schválení vozu.



Obrázek 4: Timing diagram vývoje automobilu, Zdroj: Vlastní zpracování

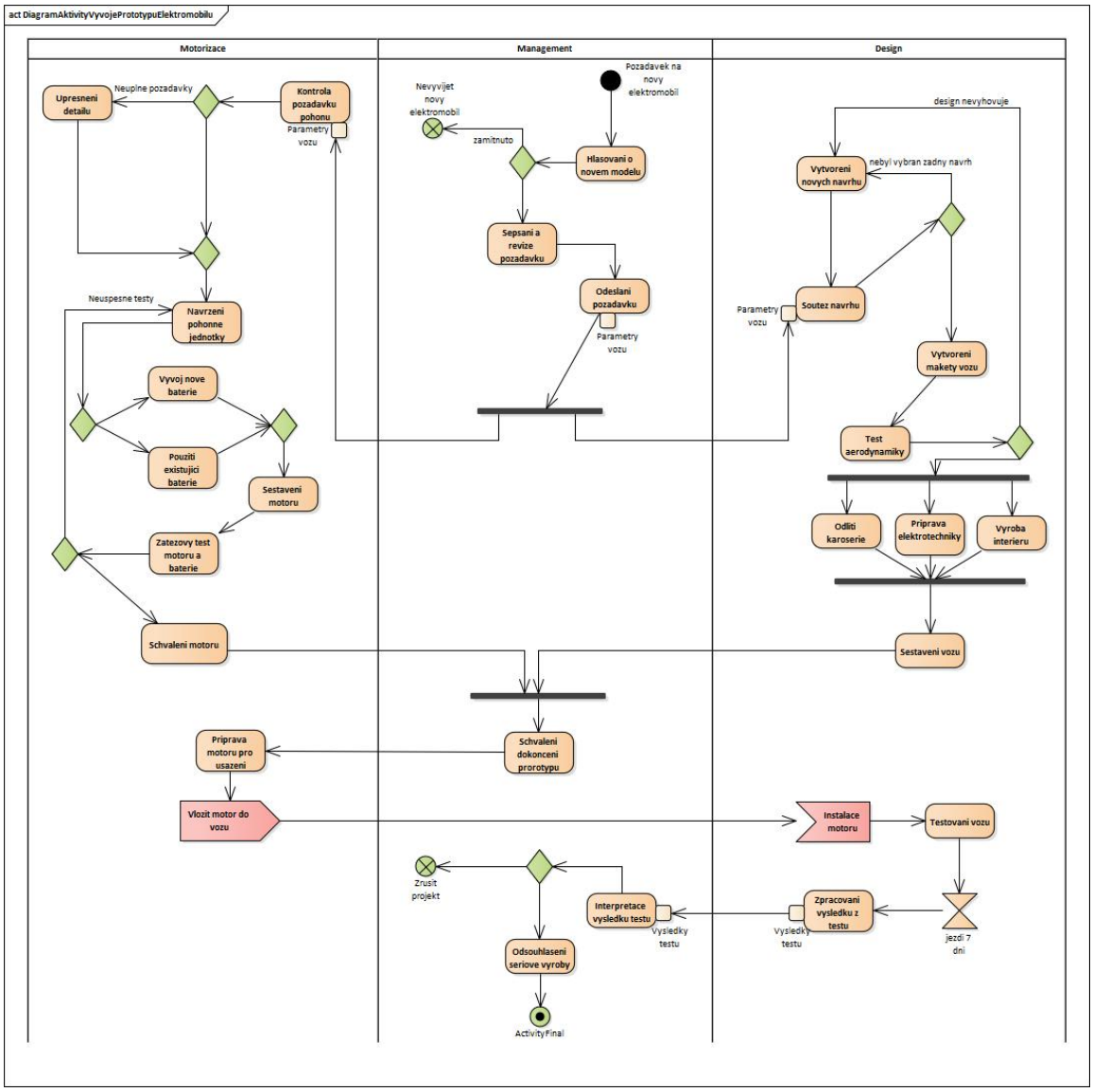
5.2.4 Activity diagram

Diagram aktivit bude, stejně jako časový diagram, vytvořen nad vývojem elektromobilu. Rozdíl bude v tom, že zatímco v časovém diagramu probíhal nejprve vývoj motorizace a až potom vývoj designu, zde budou oba vývoje probíhat paralelně, tudíž token je zároveň zaslán do více částí diagramu. Dalším rozdílným aspektem je fakt, že se již nejedná o best case scenario.

Jak je možno vidět na obrázku 5, v tomto diagramu lze již vidět rozepsaný postup jednotlivých akcí pro splnění kýžené aktivity.

Diagram je i zde rozdělen na 3 části/přepážky, které symbolizují tři oddělení - oddělení motorizace, managementu a designu. Aktivita začíná na managementu, kde z počátečního uzlu (Požadavek na nový elektromobil) dojde k přesunu tokenu na akci hlasování o novém modelu vozu. Pokud hlasování neprojde, dojde ke zrušení projektu pomoci uzlu Flow final. V opačném případě dojde k sepsání a revizi požadavků na elektromobil a následně v další akci k jejich odeslání (akce Odeslání požadavků). Zde se díky prvku fork paralelně rozvětví jak do oddělení motorizace, tak do oddělení designu.

V motorizaci dojde nejdříve ke kontrole požadavků na pohon, které byly obdrženy z parametru "Parametry vozu"zaslaného pomocí vztahu Object flow z již zmíněné akce Odeslání



Obrázek 5: Activity diagram vývoje automobilu, Zdroj: Vlastní zpracování

požadavků. Po jejich kontrole dojde k rozhodnutí, zda jsou kompletní, či nikoliv. Nekompletní požadavky projdou akcí Upřesnění detailů, poté je již možné pokračovat na Návrh pohonné jednotky. V případě, že kompletní jsou, lze se na tuto akci přesunout rovnou. Poté je nutné se rozhodnout, zda se v rozhodovacím uzlu přejde k Vývoji nové baterie, nebo k Použití existující baterie. Ať už je vybrána jakákoliv varianta, následuje samotné Sestavení motoru a po něm přichází Zátěžové testy motoru a baterie. Na rozhodovacím uzlu nastane vyhodnocení testů a pokud byly neúspěšné, je navrácena zpět na Navržení pohonné jednotky a celá tato část procesu se opakuje. Jakmile jsou testy úspěšné, přejde se na akci

Schválení motoru a po jejím splnění se token vrací zpět do oddělení managementu, kde se dostane do kontrolního uzlu join a čeká se i na druhý token ze strany vývoje designu.

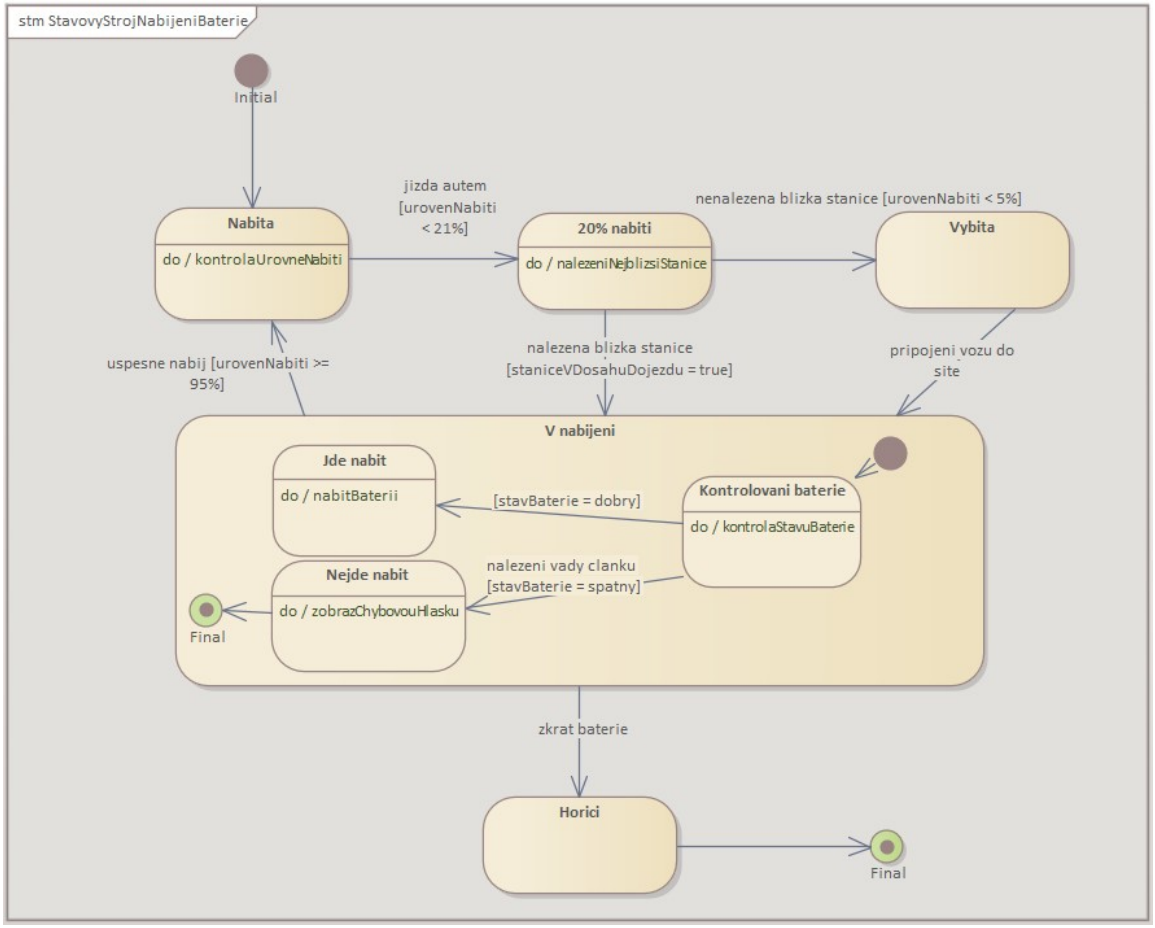
Již bylo zmíněno, že paralelně s vývojem motoru probíhá také vývoj designu. Ten v diagramu začíná Soutěží návrhů na vzhled elektromobilu, do které jsou, stejně jako u motorizace, zaslány pomocí Object flow požadované parametry vozu. Pokud v soutěži není zvolen žádný z návrhů, přejde se na akci Vytvoření nových návrhů a soutěž se opakuje. Vítězný návrh jde dále a dostane se k Vytvoření makety vozu, která následně projde akcí Test aerodynamiky. Opět zde mohou nastat 2 situace - tou první je nevyhovující design, zjištěný díky testu. V tu chvíli nastává návrat k Vytvoření nových návrhů a znovu probíhá soutěž. Pokud Test aerodynamiky proběhne úspěšně, může již dojít k akcím Odlití karoserie, Příprava elektrotechniky a Výroba interiéru. Tyto tři akce probíhají současně, jsou tudíž propojeny s forkem a joinem. Po splnění všech těchto akcí je již možné přejít k Sestavení vozu a poté k přesunu druhého tokenu zpět do join uzlu v managementu.

Zde nastává finální část diagramu, kdy dojde ke Schválení dokončení prototypu a k přesunu tokenu do oddělení motorizace. Proběhne Příprava motoru pro usazení, po níž je zaslán signál Vložit motor do vozu, který přijme oddělení Design a dojde k události Instalace motoru. Následuje Testovaní vozu, spojené s časovou událostí, během které dochází během 7 dnů k jízdním testům vozu. Informace získané během tohoto týdne jsou předány do další akce Zpracování výsledků testu. Samotné výsledky testů jsou objektovým tokem zaslány managementu, kde jsou zpracovány v Interpretaci výsledků testu. V tomto bodě dochází k poslednímu rozhodnutí o osudu prototypu. Pokud jsou výsledky špatné a neuspokojivé, dojde ke zrušení celého projektu. V opačném případě dojde v managementu k akci Odsouhlasení sériové výroby a celá aktivita je úspěšně zakončena.

5.2.5 State machine diagram

Pro diagram stavového stroje byla vybrána aktivita nabíjení baterie, kde jednotlivé stavy diagramu znázorňují stavy baterie. To lze vidět na obrázku 6.

Počáteční uzel značí novou baterii, která již z výroby přichází nabita. Prvním stavem baterie je tedy stav "Nabitá". Ten má v sobě jednu operaci do/kontrolaUrovneNabiti, která odpovídá svému názvu a kontroluje procentuální nabití. Pro přesun do dalšího stavu slouží trigger "jízda autem"společně s hlídacím stavem (guard condition) přechodu. Ten má pod-



Obrázek 6: State machine diagram baterie a jejího nabíjení, Zdroj: Vlastní zpracování

mínku [urovenNabiti $< 21\%$] a pokud je splněna, je stav změněn na "20% nabití", ve kterém se nachází operace do/nalezeniNejblizsiStanice. Z něj je možné přejít do dalších 2 stavů. Jedním z nich je "Vybitá", do kterého je možné se dostat díky nenalezení blízké stanice, úroveň nabití přitom musí klesnout pod 5% jak je možné na obrázku vidět u hlídacího stavu přechodu. Z tohoto stavu lze přejít jen do stavu "V nabíjení"pomocí připojení vozu do sítě. Druhým stavem, do kterého lze přejít ze 20% je "V nabíjení". Je toho dosaženo triggerem "nalezena blízká stanice"a hlídacím stavem [staniceVDosahuDojezdu = true]. Stav "V nabíjení"je složený stav, obsahuje tedy uvnitř více stavů. Počátečním stavem je zde "Kontrolování baterie", jež obsahuje operaci do/kontrolaStavuBaterie. Podle toho, jak na tom baterie je, může dojít ke dvěma situacím. Pokud je nalezena vada článku přejde se díky hlídací kondici [stavBaterie = spatny] do stavu "Nejde nabít". Jeho operací je do/-zobrazChybovouHlasku, poníž se přejde do finálního stavů baterie, jelikož je rozbitá a tím

pádem u konce své životnosti. Pokud je baterie v pořádku, přejde díky kondici [stavBaterie = dobry] na stav "Jde nabít", kde se díky operaci do/nabijBaterii začne nabíjet. Pokud proběhne úspěšné nabití baterie a je splněná podmínka [urovenNabiti >= 95%], přejde baterie zpět do stavu "Nabitá". Kdykoliv během stavu "V nabíjení"však může dojít ke zkratu baterie a přejde se do stavu "Hořící", ze kterého se baterie dostane do druhého finálního stavu, jelikož došlo ke zničení baterie.

5.3 BPMN

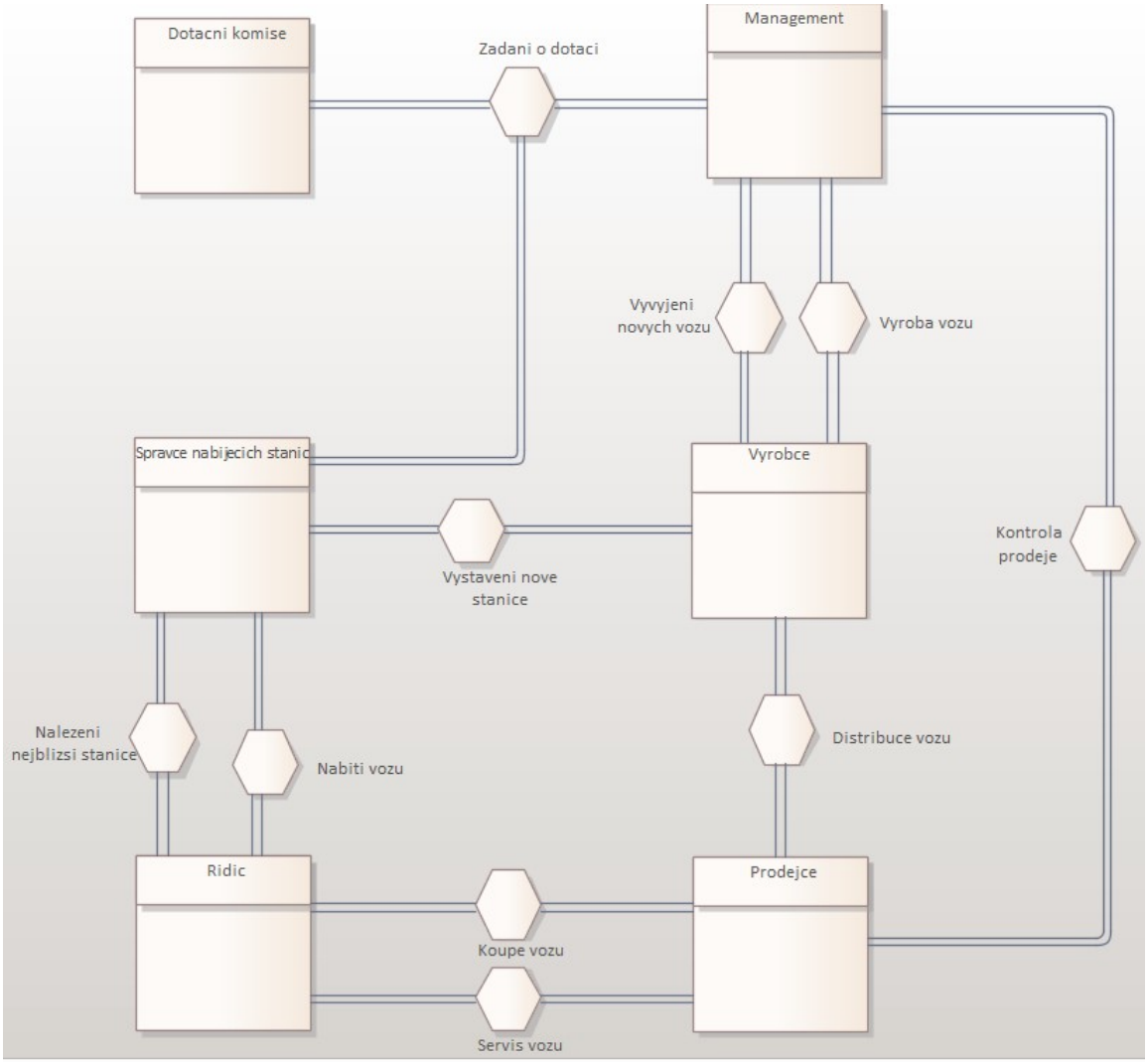
5.3.1 Conversation diagram

Diagram konverzace je soustředěn na vizualizaci množiny zpráv (tedy konverzací) mezi dvěma a více účastníky. Zde vytvořený diagram na obrázku 7 zobrazuje konverzaci účastníků procesů v rámci elektromobility.

Nachází se zde 6 účastníků, jež spolu komunikují. Jsou jimi Dotační komise, Management (zamýšleno jako část firmy rozhodující o její budoucnosti), Správce nabíjecích stanic, Výrobce (pro zjednodušení a přehlednost diagramu je v tomto účastníku zahrnuta jak výroba, tak vývoj nových modelů), Prodejce a Řidič.

Management dané firmy může podat žádost o dotaci na Dotační komisi. Dále komunikuje s prodejcem, kde probíhá Kontrola prodeje, tedy podávání reportů o úspěšnosti jednotlivých modelů na trhu. Následují konverzace s Výrobcem a jsou jimi Výroba vozů a Vyvíjení nových vozů. Poslední konverzace Managementu probíhají se Správcem nabíjecích stanic, kde může probíhat domluva o Vystavení nové stanice. Stejně jako Management, může také Správce zažádat Dotační komisi o dotace. Další konverzace správce probíhají s Řidičem. Jedná se o Nabití vozu a Nalezení nejbližší nabíjecí stanice (zde by mohlo být řešeno buďto pomocí GPS či mobilní aplikace). Řidič může mít konverzace s Prodejcem, a těmi jsou Koupě vozu a Servis vozu. Prodejce už má potom jen jedinou další konverzaci, a to s Výrobcem ohledně Distribuce nových vozů.

Podobně jako u UML Use case diagramu lze také o modelu konverzace říci, že se jedná o high level pohled na modelovaný systém z hlediska komunikace jednotlivých aktérů.

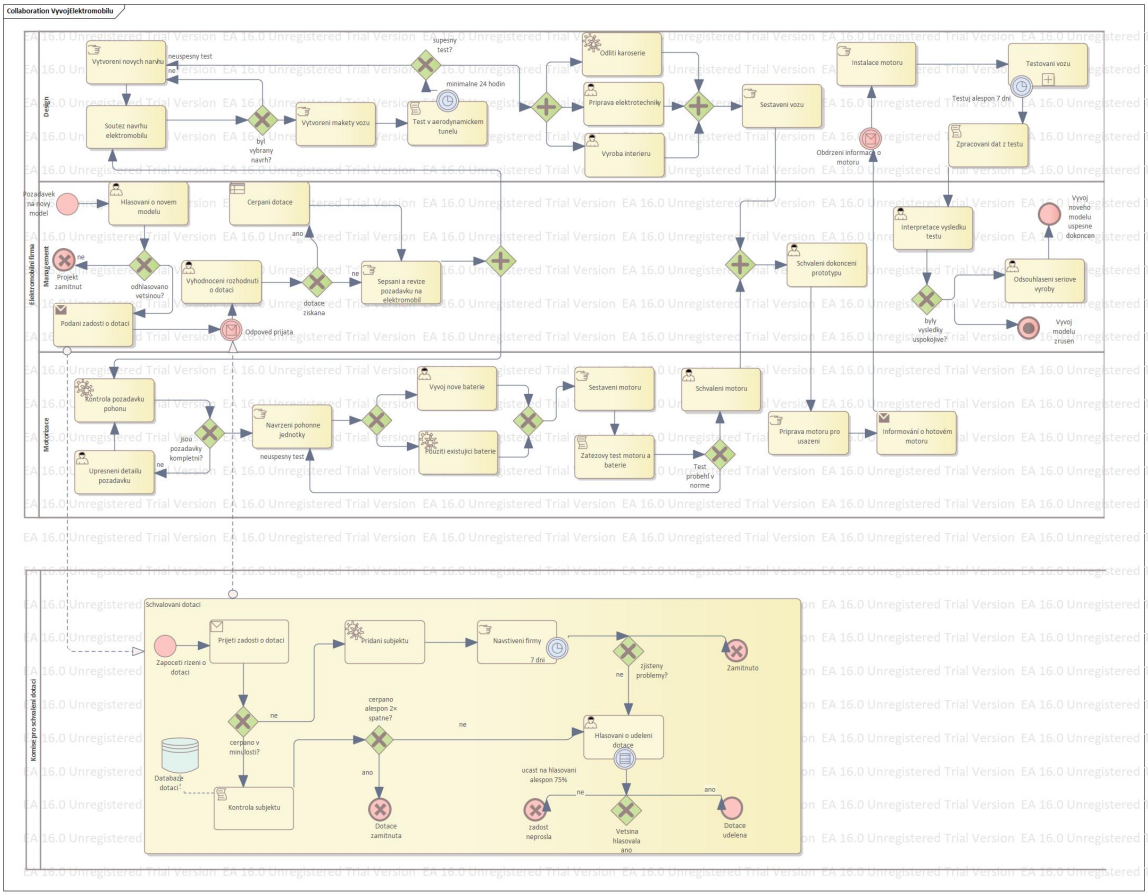


Obrázek 7: Conversation diagram účastníku elektromobility, Zdroj: Vlastní zpracování

5.3.2 Collaboration diagram

Diagram kolaborace je využíván pro modelování složitějších či delších business procesů, ve kterých je nutná konverzace mezi více účastníky. V tomto diagramu se opakuje scénář vývoje elektromobilu, avšak s tím rozdílem, že výrobce komunikuje také s externí entitou, viz obrázek 8.

Princip tohoto diagramu je podobný jako u activity diagramu v UML, nicméně je tu, kromě již zmíněné externí komunikace mimo aktivitu samotnou, poměrně velká řada odlišností či novinek. Každé akci je možné přiřadit její typ, který už na první pohled prozradí, jakým způsobem má být prováděna. To už v prvopočátku může usnadnit vývoj a navazující



Obrázek 8: Collaboration diagram vývoje nového modelu elektromobilu, Zdroj: Vlastní zpracování

implementaci systému, díky detailnějšímu popisu jednotlivých úkolů a odpadnutí nutnosti každou akci mnohdy zdlouhavě prodiskutovávat.

Mezi použité varianty tasků v modelu patří:

- Abstract Task, což je obyčejný úkol, který není nijak dále specifikován.
- User Task oproti tomu již říká, že je u procesu nutný člověk, který však při jejím konání spolupracuje se systémem nebo softwarovou aplikací, jež mu napomáhá. Může to být například interpretace výsledků testů prototypu, kdy je nutné podle určitých postupů a platných předpisů určit, zda vyvíjený prototyp odpovídá požadovaným standardům.

- Service Task je akcí, kterou provede uřčitá automatizovaná aplikace nebo stroj. V modelu je použita například pro automatické přidání subjektu do databáze dotací při první žádosti.

- Script Task bývá vykonávána systémem business procesu, pokud je pro danou akci vytvořen skript v jazyku, který je systémem spustitelný. Takový skript by byl použit například pro zátěžové testování motoru. Zde je potřeba, aby proběhly jednotlivé fáze testu pro přesné zjištění jeho limitů, výkonu a podobně.

- U Manual Task se očekává, že bude provedena bez jakékoliv pomoci ze strany systému či podobných asistentů, vykonává ji pouze samotný člověk. Pod tím si lze v modelu představit například manuální instalaci motoru do vozidla.

- Send Task a Receive Task jsou používány pro zasílání zpráv (nejčastěji) externím a jejich přijímání od externích účastníků business procesu. V procesu na obrázku slouží send k zaslání zprávy dotační komisi a receive k obdržení odpovědi.

Dalším pomocníkem pro zjednodušení budoucí implementace mohou být intermediate events. Ty bývají nejčastěji připojeny na hranách akcí, z nichž poté pokračuje tok do dalších částí diagramu. Mnohdy se vztahují k určitým podmínkám (tzv. Conditional event) či časovým omezením (Timer). Jedním příkladem podmínky může být akce "Hlasování o udělení dotace", na jejíž hraně je podmínka, že účast na hlasování musí být alespoň 75% . Pokud toto není splněno, nelze přejít z této akce dál. Událost s časovým omezením funguje tak, že pokud není splněno, že ku příkladu testování vozu bude probíhat alespoň 24 hodin, nelze se přesunout na další akci či rozhodovací uzel. Mimo připojení na hrany je možné "mezilehlé"události použít i samostatně. Pro příklad lze uvést přijetí zprávy systémem (mimo akce), předání signálu nebo více časových událostí připojených k rozhodovacím uzlům.

Co se týče samotného diagramu, kromě již sepsaných vylepšení v něm přibyl také externí účastník. Je jím Komise pro schvalování dotací. S tou komunikuje Management pomocí mechanismu toku zprávy (společně se Send a Receive úkoly), který je používán pro komunikaci mimo pooly. Dotační komise jednu složenou aktivitu, kterou je Schválení dotace. Ta začíná receive taskem Přijetí žádosti o dotaci, z níž pokračuje tok do rozhodovacího uzlu, ve kterém se zjišťuje, zda žadatel o dotaci již v minulosti někdy podporu čerpal. V případě, že žádá poprvé, dojde k automatickému přidání subjektu do systému, a poté proběhne návštěva firmy komisí, aby se zjistilo, zda je vše v pořádku a odpovídá předpisům. Kontrola trvá 7

dní a když proběhne bez problémů, je možné se přesunout ke hlasování o udělení dotace. Pokud jsou však nalezeny problémy nebo zjištěny nesrovnalosti, žádost se zamítá. Za podmínky, že firma už v minulosti dotace čerpala, jsou z databázi načteny záznamy o průběhu čerpání. Následuje skriptovaná kontrola zjištěných informací, která vede ke zjištění, zda čerpání neproběhlo již alespoň 2× za špatných okolností. Je-li tomu tak, žádost se automaticky zamítá a tok schvalování zde končí. V opačném případě lze také přejít na Hlasování o udělení dotace. Hlasování může proběhnout pouze pod podmínkou, že se ho zúčastní minimálně 75% všech hlasujících. Pokud hlasuje většina proti, žádost je zamítnuta a aktivita končí. Hlasuje-li však většina pro návrh, je dotace udělena a aktivita končí úspěšně.

Výsledek schvalování je zaslán zpět Managementu, kde vyhodnocen a buďto dojde k přesunu na Sepsání požadavku na alektromobil (pokud nebyla dotace získána) nebo na akci Čerpání dotace (ta spadá pod typ Business Rule Task, tedy vstup do ní i výstup z ní je řízen určitými pravidly), a po jejím splnění může také pokračovat na Sepsání požadavků. Další postup diagramem se nese v podobném stylu vývoje motorizace a designu, jako Activity diagram.

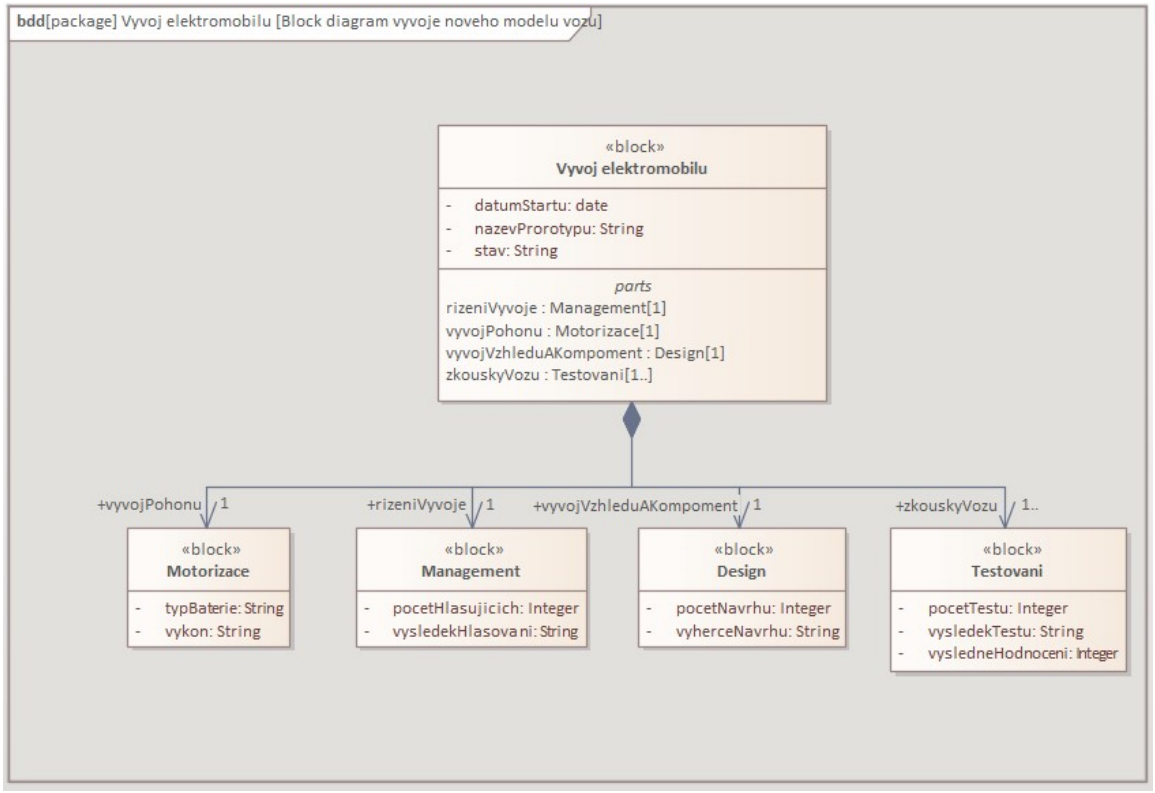
V porovnání s Activity diagramem vývoje elektromobilu je u modelování business procesu v BPMN výrazný rozdíl v zápisu jednotlivých akcí a podmínek přesunů z nich. Obecně lze říct, že BPMN je přehlednější, lze v něm zaznamenat více detailů, což usnadňuje a urychluje pochopení modelovaného procesu. Také je možné přidat externí účastníky takovým způsobem, aniž by to zhoršilo čitelnost diagramu. Vzhledem k úrovni, s jakou jde modelovat podrobnosti jednotlivých aktivit či akcí, lze konstatovat, že BPMN může oproti UML activity diagramu ušetřit čas. To jak vývojářům tak analytikům a manažerům, a tím pádem může pomoci snížit nejen výdaje na vývoj softwaru, ale později také na běh celého procesu (díky kvalitnímu a detailnímu návrhu).

5.4 SysML

5.4.1 Block Definition Diagram

Podobně jako class diagram, ani Block Definition Diagram není diagramem chování. Nicméně lze z něho jednoduše vyčíst spolupracující, či na sebe napojené části systému. Diagram na obrázku 9 představuje pouze výseč systému, na který by tato byly navázány. Přesto na něm lze ukázat, jakým způsobem mohou být jednotlivé bloky dále rozloženy. Kromě bloků sa-

motných lze v BDD modelovat také na sebe navazující aktivity, jak by mohly být následně vytvořeny v diagramech chování.



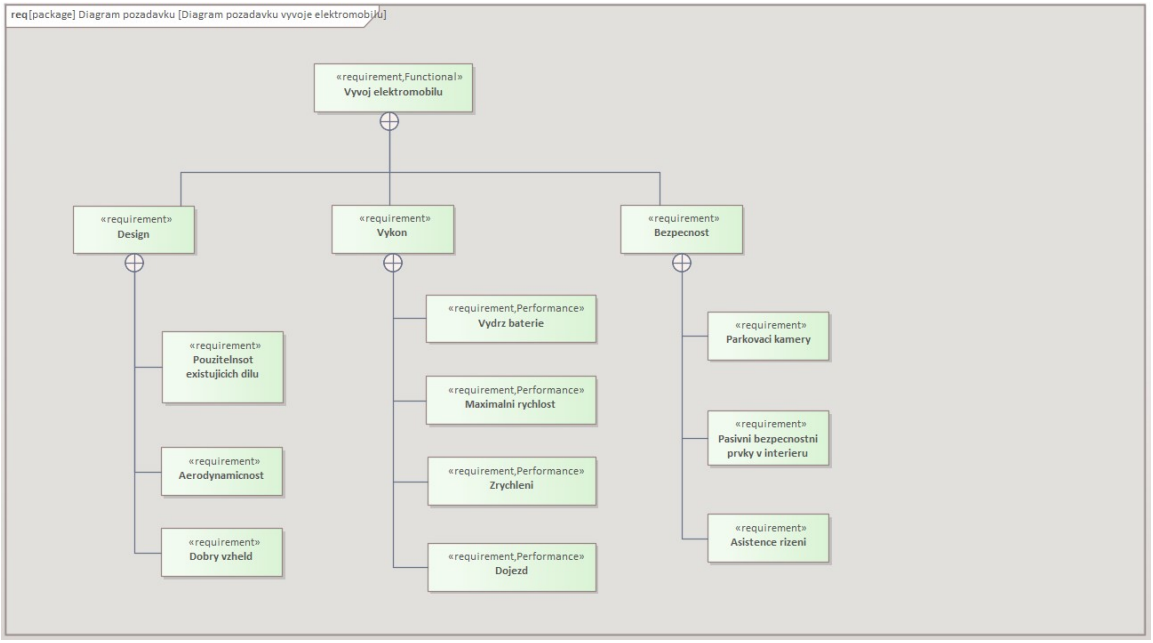
Obrázek 9: Block Definition Diagram vývoje nového modelu elektromobilu, Zdroj: Vlastní zpracování

Vytvořený diagram bloků ukazuje aktivitu vývoje elektromobilu. Na "hlavní"blok jsou pomocí kompozice napojeny jeho části (parts), které v tomto případě představují jednotlivé účastníky procesu vývoje. Jsou jimi opět oddělení, a to konkrétně Motorizace, Management, Design a Testování. Každý blok může mít své vlastní atributy a případně také operace.

5.4.2 Requirments Diagram

Diagram požadavků je takzvaným cross-cutting diagramem, tedy může zasahovat jak do diagramů strukturálních, tak do diagramů chování. Na obrázku 10 je ukázán rozpis požadavků pro vývoj nového elektromobilu.

Na první pohled je z diagramu jasná hierarchická struktura, kde je vývoj rozdělen na tři nadřezené požadavky. Prvním nadřazeným požadavkem je Design requirment, do kterého spadají nároky na Použitelnost již existujících dílů (v rámci úspor), na Aerodynamičnost



Obrázek 10: Requirments Diagram vývoje nového modelu elektromobilu, Zdroj: Vlastní zpracování

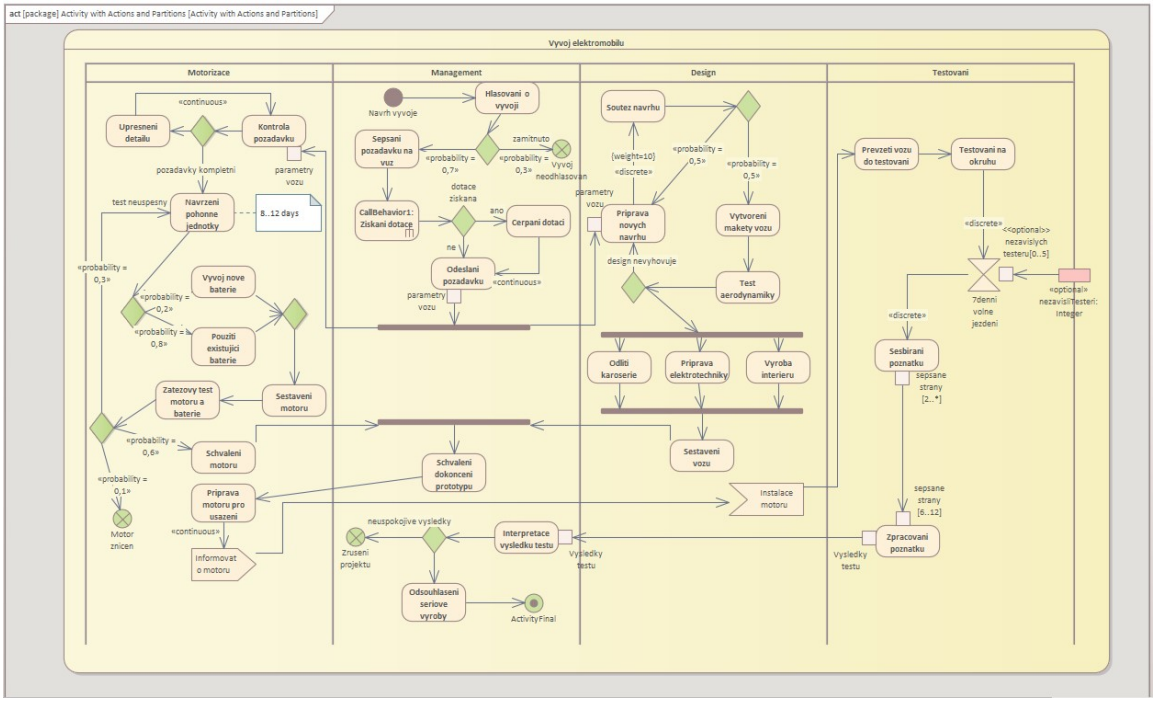
designu a také požadavek na Dobrý vzhled (kvůli prodejnosti). Dalším výše postaveným požadavkem je Výkon. Do něj patří Výdrž baterie, Maximální rychlost, Zrychlení a Dojezd. Posledním ze trojice je Bezpečnost. Zde je kladen důraz na Asistence (asistenty) řízení, Parkovací kamery a Pasivní bezpečnostní prvky v interiéru (zejména airbagy).

Tento diagram jde jednoduše spojit jak s blokovým diagramem Vývoje vozu, tak s diagramem aktivit. V něm se vyskytuje akce "Sepsání požadavků na vůz", a Následné "Odeslání požadavků". Požadavky se poté předávají jako parametr na oddělení Motorizace a oddělení Designu, kde jsou zapracovány do návrhů vozu.

5.4.3 SysML Activity Diagram

Jediným diagramem chování, který byl v notaci SysML změněn, či upraven oproti UML, je diagram aktivit. Oproti UML poskytuje způsob, jakým lze označit, zda kontrolní tok mezi uzly proběhne okamžitě, či může nějakou chvíli trvat. Jak lze vidět na snímku 11 diagramu, některé z kontrolních toků mají označení continuous či descrete.

Vazba se stereotypem continuous může být viděna například mezi akcemi Upřesnění detailu a Kontrola požadavku (partition Motorizace). Znamená to, že k přesun mezi těmito

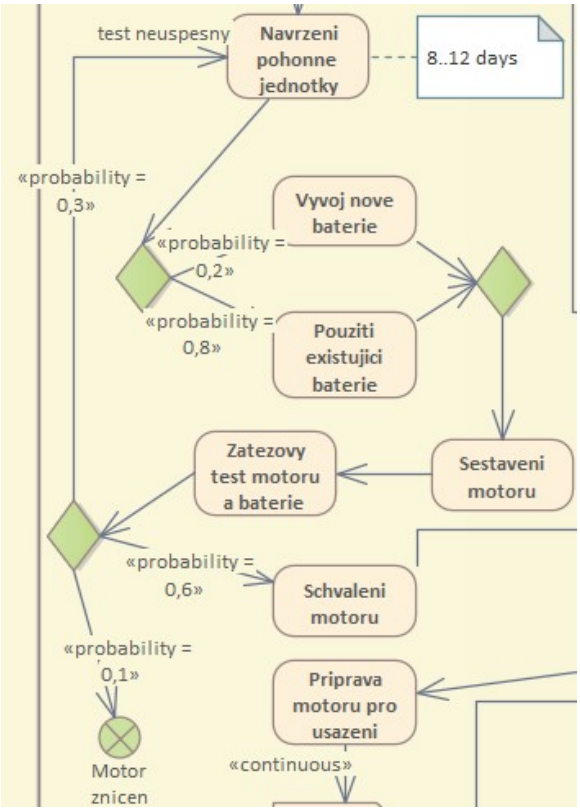


Obrázek 11: SysML activity diagram vývoje nového modelu elektromobilu, Zdroj: Vlastní zpracování

dvěma akcemi dojde prakticky okamžitě, tedy v čase, který se blíží nule. Prakticky to znamená, že jakmile jsou detaily požadavků upřesněny, dojde k jejich okamžité kontrole.

Stereotyp descrete u vazby znamená pravý opak, čas přesunem na další akci je nenulový a může nějakou chvíli trvat. V diagramu je to vidět u akce Příprava nových návrhů v Designu. Přesun od přípravy návrhů k soutěží může trvat déle, navíc je zde také podmínka { weight=10 } která udává, že do soutěže musí být zapojeno alespoň 10 návrhů (tedy musí do ní projít v danou chvíli 10 tokenů z přípravy, aby mohla být akce započata).

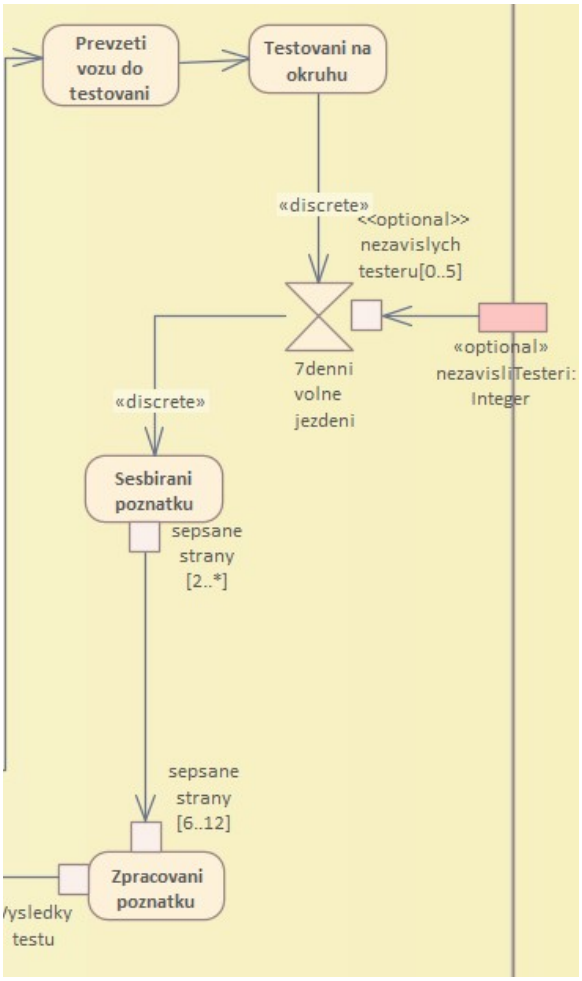
Další změnou oproti UML je možnost nastavení pravděpodobností přechodu, které mohou být vidět například na obrázku (12) detailu oddělení Motorizace z předešlého diagramu. Po navržení pohonné jednotky dochází k rozhodnutí, zda vyvinout novou baterii nebo použít již osvědčené existující baterie. Jelikož vývoj může být nákladný a zdlouhavý, je jeho pravděpodobnost pouze 0,2 (tedy 20%), zatímco pravděpodobnost použití existující baterie je 0,8 (80%). Podobně je tomu u zátěžových testů motoru. Nejvyšší šance (probability =0,6) je u úspěšného testu s následným přesunem na schválení motoru, nížší šance (0,3) pro neúspěšný test a nejnižší šance (0,1) je pro zničení motoru a následné ukončení projektu.



Obrázek 12: Detail pravděpodobnosti přechodu u Motorizace, Zdroj: Vlastní zpracování

Na obrázku 13 je ukázka použití parametrů. Externí parametr (růžový obdélník) představuje nezávislé testery z řad veřejnosti či tisku. Jak je možné vidět u vstupního parametru časové události (7 denního volného ježdění), takovýchto řidičů může být maximálně pět, a jelikož je to volitelné, nemusí také dorazit nikdo. Oproti tomu existují povinné parametry, které je možné pozorovat u Sesbírání poznatků a Zpracování poznatků. Pro to, aby bylo možné se přesunou z akce sbírání, musí být sepsány alespoň 2 strany, jejich horní limit však udán není. Vstupní parametr u zpracování je minumum 6 stran, maximum poté 12. Pokud jich není alespoň 6, akce nemůže započnout. Teprve poté může být provedena a průchod tokenu diagramem pokračuje.

Na těchto ukázkách jsou vidět vylepšení, která nastala v porovnání s UML variantou diagramu. Zvláště pro přechody (kontrolní toky) mezi aktivitami a jejich vstupními/výstupními parametry lze u SysML Activity diagramu zaznamenat mnohem větší detaily, než by bylo jinak možné. To může výrazně pomoci při simulování průchodu diagramem, jakožto také pro vytvoření lepší představy, jak spolu budou jednotlivé akce spolupracovat a zda



Obrázek 13: Detail parametrů a stereotypu optional, Zdroj: Vlastní zpracování

si budou něco zasílat (včetně multiplicit), ať už v navrhovaném systému, nebo při vývoji přidružených aplikací.

6 Výsledky

Každý ze zpracovaných diagramů má svůj vlastní účel a silně záleží na tom, jaký projekt je pomocí něj modelován. Obecně lze konstatovat, že Use Case diagram by se měl nacházet u téměř každého projektu, jelikož se jedná o základní, avšak high-level pohled na modelovaný systém či aplikaci. Lze z něj vyčíst množství přínosných informací potřebných pro vytváření dalších diagramů, a to ať už strukturálních, nebo diagramů chování. Časový diagram lze ve zmiňovaném případě použít pro orientaci, jak dlouho by maximálně takový vývoj mohl probíhat. Tento odhad může pomoci při určování, jaký časový limit by měl být nastaven jednotlivým fázím vývoje. V další variantě použití může být udáváno, jaká doba by měla být vyhrazena vyšším pozicím v managementu pro rozhodování o důležitých záležitostech v rámci vývoje. Důvodem pro to může být, aby vývoj nestagnoval a podřízená oddělení měla aktuální informace. Vzhledem k tomu, že ostatní diagramy řeší zápis času minimálně nebo vůbec, je vhodné časový diagram vytvářet i v případě práce s ostatními notacemi. V každém případě poslouží jako vhodný doplněk pro popis chování určitého procesu v čase. Diagram stavového stroje ukazuje vnitřní chování třídy nebo aktivity. V rámci modelovaného diagramu to byla třída Baterie, ve které byly ukázány její vnitřní stavy. U každého stavu je možné nastavit operace, které se během něj mohou vykonat, a také podmínky pro přechod do dalších stavů. Složené stavy mohou obsahovat více vnitřních stavů, mezi kterými mohou na základě okolností přecházet. Z pohledu přínosu v rámci baterie jsou tyto stavy důležité, jelikož hlídají například úroveň nabití nebo kontrolují zdravé baterie. Díky tomu lze vyhodnotit, zda se baterie nabíjí správně, nebo zda vznikla vada a baterii je nutné vyměnit, dříve než nastanou závažnější problémy. V případě modelování vývoje vozu by v tomto typu diagramu bylo možné modelovat například stavy vývoje pohonné jednotky, kompletace návrhů a stavby karoserie prototypu, nebo stav testování sestrojeného prototypu. Tyto stavy by následně mohly posloužit coby dokumentace kompletního vývoje a vést ke zlepšení efektivity budoucích vývojů. Diagram aktivit poskytl zápis postupu akcí pro vyvinutí nového modelu elektromobilu. Z hlediska UML v porovnání s ostatními notacemi se jedná o diagram, ve kterém jsou znázorněny prvky spíše v jednodušší formě. Avšak už zde je možné předat velké množství informací o tom, jak by měl být tento proces v systému v budoucí době implementován. Je možné ho také využít jako jakousi předlohu, ze které se bude vycházet u dalších diagramů, které mají modelovat obchodní procesy. Tako-

vými diagramy mohou být ku příkladu BPMN collaboration diagram, nebo SysML Activity diagram, který UML verzi obohacuje o nové možnosti zápisu detailů.

V notaci BPMN byly modelovány dva diagramy. Prvním byl diagram konverzace, jehož role v modelu je přinést vyšší pohled na vytvářený systém ze strany procesů a účastníků, kteří s nimi pracují. Jedná se o přehledný a lehce pochopitelný model, ve kterém spolu komunikují vždy dva a více účastníků, kteří mají společnou "konverzaci". Lze z něj jednoduše vypozorovat, jak jsou jednotliví uživatelé propojeni a jak silná provázanost mezi nimi existuje. Obecně lze říct, že diagram konverzace je zjednodušenou verzí kolaboračního diagramu. Může modelovat pohled na celý systém (jako v případě vytvořeného a dříve zmíněného diagramu), nebo také jen samostatné procesy, jako právě vývoj elektromobilu. Pomáhá utvořit jakousi myšlenkovou mapu před samotným detailnějším modelováním celého projektu či vybraných procesů. Druhým diagramem je kolaborace. Dalo by se říct, že se jedná o activity diagram posunutý na novou úroveň. Lze v něm lépe prezentovat, jakým způsobem mají být akce prováděny, jde lépe naznačit výstupní podmínky z akcí (intermediate events), aby mohl být token přesunut dál. Z celkového hlediska je vhodnější pro modelování business procesů, než jeho protějšek v UML. Efektivnější označení akcí a aktivit pomáhá ušetřit čas, který by byl za jiných podmínek potřeba pro další komunikaci mezi zadavatelem a analytikem, či přímo vývojářem. Lze tak snadno říct, že BPMN v tomto ohledu značně zefektivňuje analýzu a modelování obchodních procesů a díky tomu (mimo jiné) pomáhá také ušetřit výdaje. Modelování v SysML bylo započato, ač ne diagramem chování, Block

Definition diagramem. Na něm byl lehce ukázán rozdíl v modelování struktur oproti UML. Requirments diagram, neboli diagram požadavků, byl navržen tak, aby reflektoval základní potřeby pro vývoj nového modelu elektromobilu. Je to jeden z diagramů, který je na pomezí strukturálních a behaviorálních diagramů. Modelovaný diagram vhodně doplňuje jak Activity diagramy (UML, SysML), tak Collaboration diagram (BPMN). Ve všech těchto diagramech vytvořených v ohledu k vývoji elektromobilu, se zasílají specifikace, které by měl prototyp splňovat. Modelovaný diagram požadavků by mohl být tím pádem provázaný s danými akcemi, které se o jejich zasílání a přijímání starají. Posledním diagramem vytvořeným v SysML byl diagram aktivit. V porovnání s UML funguje v základu na stejných principech, nicméně je vylepšen o několik funkcí, které z něj dělají lepšího kandidáta pro modelování systémů. Mezi tato vylepšení patří patří continuous×discrete control flow,

pravděpodobnosti u rozvětvených cest (tedy na rozhodovacích uzlech), optional parametry a další. SysML diagram aktivit může stále přijít vhod i v případě, že již byl zhotoven kolaborační diagram v BPMN. Důvodem pro to může být jeho využitelnost při simulování průchodu danou aktivitou. Zejména pravděpobonosti přechodu mohou při simulacích naznačit, jak spolehlivý daný business proces může být a zároveň určit případnou procentuální chybovost či neúspěšnost.

Nakonec je tedy možné konstatovat, že pro potřeby modelování obchodního procesu Vývoje elektromobilu se nejlépe hodí BPMN s jeho diagramem kolaborace případně také konverzace. Diagram konverzace bývá vhodné vytvořit dříve, než diagram kolaborace, jelikož může pomoci odhalit uživatelské interakce, či interakce aktivit a akcí, které by jinak mohly být vynechány. Neobejde se ovšem bez některých diagramů z UML, zejména Use Case a class. Use case diagram je ideální vytvářet při každém projektu, ve kterém figuruje více aktéru, přičemž každý z nich má na sebe navázaných několik procesů, které navíc mohou být využívány i dalšími aktéry. Ostatní UML diagramy je možné modelovat dle potřeby, které vyvstanou z analýzy projektu, případně na požadavky zákazníků. Jedním z nich může být časový diagram. Ten je v projektech vhodným doplňkem pro modelování změn v průběhu času, a to ať už ve vybrané třídě, nebo aktivitě. Lze ho také využít pro zobrazení životních cyklů. Ze SysML je vhodné využít minimálně diagram požadavků. Ten může pomoci zapsat určité poznatky, což se v již zmíněných notacích provádí s velkými obtížemi a je tedy jednodušší "sáhnout"do této notace. SysML activity diagram může také přijít vhod, zejména, pokud může v procesu například nastat několik nechtěných situací, se kterými se však dá dopředu počítat. Ty se dají naznačit pomocí pravděpodobností přechodů a mohou pomoci při simulacích modelu. Těmi lze zjistit, jak sppolehlivý bude daný proces, případně jakou bude mít chybovost. Všechny notace mají své plusy i mínusy a ne každá z nich lze využít libovolný typ projektu. Dalo by se tedy shrnout, že mnohdy je nejlepší využít kombinace více modelovacích jazyků či notací, aby bylo možné daný projekt pojmout ze skutečně kompletního hlediska, pokud je k tomu příležitost, čas a finance.

Pro modelování procesu vývoje elektromobilu by tedy mohla být vhodná kombinace použitých diagramů v následujícím zastoupení. Z UML je ideální zvolit diagram případů užití, class diagram, časový diagram a kombinace několika stavových diagramů pro přenesený popis postupu prací na prototypu a jeho testování. V BPMN diagram konverzace

a nejpodstatnější diagram pro modelování tohoto procesu, kolaborační diagram. Ze SysML by byl zvolen diagram požadavků a pro případné simulace průchodu obchodním procesem také diagram aktivit.

7 Závěr

Diplomová práce se zabývala tématem modelování diagramů chování napříč notacemi UML 2.0, BPMN 2.0 a SysML. Tyto notace a modelovací jazyky byly rozebrány a popsány v teoretické části práce. U jednotlivých modelovacích notací byla popsána historie jejich vývoje a zároveň popsány diagramy, které do nich patří a jaké je jejich použití. Samotné diagramy chování byly analyzovány ve větší míře v samostatné kapitole. Zde došlo k jejich podrobnějšímu prozkoumání, včetně detailního popisu prvků, ze kterých se skládají. Cílem této práce bylo vytvoření diagramů chování v notacích UML, BPMN a SysML, jejich rozbor a u určitých diagramů porovnání a popis jejich podobností či rozdílů. To bylo provedeno v praktické části, kde bylo nejprve představeno téma, kterým se bude modelování zaobírat. Tím byla udržitelnsot a elektromobilita. Poté již následoval samotný popis diagramů vytvořených v rámci tohoto tématu. Nejprve byly představeny diagramy v Unified Modeling Language, které nastínily základní směr těchto modelů, a také business proces, o němž bude většina diagramů pojednávat. Následovala notace BPMN, kde byly například u diagramu kolaborace popsány změny, vylepšení a nové prvky oproti UML diagramu aktivit. Poslední notací, ve kterých byly modely na téma elektromobility vytvořeny, bylo SysML. Došlo k nastínění modelování struktury u této notace a byl představen nový diagram v této notaci, a to diagram požadavků. Posledním výtvorem byl diagram aktivit, kde byl předem vytvořený a popsaný UML diagram obohacen o vylepšení, která do něj byla přidána v rámci notace SysML. Byla uvedena jeho výhoda oproti UM variantě, či dokonce kolaboračnímu diagramu, kterou je simulace průchodu aktivitou. Ve výsledcích byly shrnuty modelované diagramy, o co v nich šlo a jaký by mohly mít přínos v rámci modelovaného procesu. Byly vysvětleny důvody, který typ diagramu se více hodí pro modelování dané situace a čím může oproti ostatním pomoci ohledně budoucí implementace systému. Ke konci byl vybrán nejvhodnější typ diagramu pro vytvoření procesu vývoje elektromobilu, kterým se stal kolaborační diagram v BPMN. Bylo ovšem uznáno, že pro zhotovení kompletního modelu mnohdy jediná notace či jazyk nestačí, a je proto vhodné z každé z nich použít vybrané diagramy dle potřeb modelovaného systému. Pro vývoj elektrického vozidla byly nakonec vypsány diagramy, které by mohly posloužit pro celistvé modelování tohoto obchodního procesu.

Literatura

[1] Uml Diagrams vs Sysml Diagrams. [Cit. z 21.9.2022]. Dostupné z: https://www.lucidchart.com/blog/uml-diagrams-vs-sysml-diagrams

[2] UML Diagrams: Everything You Need to Know to Improve Team Collaboration. [Cit. z 5.11.2022]. Dostupné z: https://www.bluescape.com/blog/uml-diagrams-everything-you-need-to-know-to-improve-team-collaboration

[3] History of UML: Methods and Notations. [Cit. z 5.9.2022]. Dostupné z: https://sourcemaking.com/uml/basic-principles-and-background/history-of-uml-methods-and-notations

[4] Software engineering: Software crisis. [Cit. z 6.9.2022]. Dostupné z: https://www.geeksforgeeks.org/software-engineering-software-crisis/

[5] What is UML? Everything you need to know about Unified Modeling Language. [Cit. z 10.9.2022]. Dostupné z: https://www.gliffy.com/blog/what-is-uml-everything-you-need-to-know-about-unified-modeling-language

[6] UML Diagrams: History, Types, Characteristics, Versions, Tools. [Cit. z 10.9.2022]. Dostupné z: https://www.guru99.com/uml-diagrams.html#1

[7] Difference between Object Model Diagram and Class Diagram. [Cit. z 12.9.2022]. Dostupné z: https://www.ibm.com/support/pages/difference-between-object-model-diagram-and-class-diagram-rational-rhapsody

[8] What is Package Diagram. [Cit. z 12.9.2022]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/

[9] What is Component Diagram. [Cit. z 12.9.2022]. Dostupné z: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/

[10] Deployment Diagram Tutorial. [Cit. z 12.9.2022]. Dostupné z: https://creately.com/blog/diagrams/deployment-diagram-tutorial/

[11] Timing Diagrams. [Cit. z 12.9.2022]. Dostupné z: https://www.uml-diagrams.org/timing-diagrams.html

[12] The Sequence Diagram. [Cit. z 12.9.2022]. Dostupné z: https://developer.ibm.com/articles/the-sequence-diagram/

[13] Communication Diagrams. [Cit. z 12.9.2022]. Dostupné z: https://www.uml-diagrams.org/communication-diagrams.html

[14] What is Business Process Modeling Notation. [Cit. z 14.9.2022]. Dostupné z: https://www.lucidchart.com/pages/bpmn

[15] *Business Process Model and Notation (BPMN), Version 2.0*. Object Management Group, 2011, 112–114 pp. Dostupné z: https://www.omg.org/spec/BPMN/2.0/PDF

[16] *Business Process Model and Notation (BPMN), Version 2.0*. Object Management Group, 2011, 306–308 pp. Dostupné z: https://www.omg.org/spec/BPMN/2.0/PDF

[17] *Business Process Model and Notation (BPMN), Version 2.0*. Object Management Group, 2011, 315–316 pp. Dostupné z: https://www.omg.org/spec/BPMN/2.0/PDF

[18] *Business Process Model and Notation (BPMN), Version 2.0*. Object Management Group, 2011, 109–112 pp. Dostupné z: https://www.omg.org/spec/BPMN/2.0/PDF

[19] *Business Process Model and Notation (BPMN), Version 2.0*. Object Management Group, 2011, 124–126 pp. Dostupné z: https://www.omg.org/spec/BPMN/2.0/PDF

[20] What is Sysml. [Cit. z 22.9.2022]. Dostupné z: https://www.omgsysml.org/what-is-sysml.htm

[21] Who Created Sysml. [Cit. z 23.9.2022]. Dostupné z: https://sysml.org/sysml-faq/who-created-sysml.html

[22] McLaughlin, P. G. W. D., B. *Head First Object Oriented Analysis and Design*. O'Reilly Media, Inc., 2006, 297 pp., iSBN: 978-0596008673.

[23] Jim Arlow, I. N. *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*. Addison-Wesley Professional, second edition, 2006, 67–93 pp., iSBN: 978-0321321275.

[24] Jim Arlow, I. N. *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*. Addison-Wesley Professional, second edition, 2006, 283–298 pp., iSBN: 978-0321321275.

[25] Jim Arlow, I. N. *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*. Addison-Wesley Professional, second edition, 2006, 439–448 pp., iSBN: 978-0321321275.

[26] Jim Arlow, I. N. *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*. Addison-Wesley Professional, second edition, 2006, 449–453 pp., iSBN: 978-0321321275.

[27] Anders Grauers, M. K., Steven Sarasini. WHY ELECTROMOBILITY AND WHAT IS IT? ISBN 978-91-980973-1-3. Dostupné z: `https://publications.lib.chalmers.se/records/fulltext/211430/local_211430.pdf`

[28] Roman Chinoracky, T. C., Natalia Stalmasekova. Trends in the Field of Electromobility—From the Perspective of Market Characteristics and Value-Added Services: Literature Review. 2022, [Cit. z 10.11.2022]. Dostupné z: `https://doi.org/10.3390/en15176144`

[29] Trends and developments in electric vehicle markets. [Cit. z 11.11.2022]. Dostupné z: `https://www.iea.org/reports/global-ev-outlook-2021/trends-and-developments-in-electric-vehicle-markets`

Přílohy

1) DP-Elektromobilita-Prochazka-Vilem.qea
   - obsahuje modelované diagramy
   - vytvořeno v Enterprise Architect v16

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2020/2021

Studijní program: Aplikovaná informatika
Forma studia: Prezenční
Obor/kombinace: Aplikovaná informatika (ai2-p)

Podklad pro zadání DIPLOMOVÉ práce studenta

Jméno a příjmení: **Bc. Vilém Procházka**
Osobní číslo: **I2000053**
Adresa: Milady Horákové 261, Hradec Králové – Třebeš, 50006 Hradec Králové 6, Česká republika

Téma práce: Diagramy chování a jejich praktické použití napříč notacemi UML, SysML, BPMN
Téma práce anglicky: Behavioural diagrams and their use across the notations UML, SysML and BPMN

Vedoucí práce: doc. Ing. Hana Tomášková, Ph.D.
Katedra informačních technologií

Zásady pro vypracování:

Osnova:

1. Úvod
2. Modelovací notace
3. Analýza diagramů chování
4. Využití diagramů chování
5. Výsledky
6. Závěr

Seznam doporučené literatury:

Sommerville, I. (2013). *Softwarové inženýrství*. Computer Press, Albatros Media as.
Arlow, J., & Neustadt, I. (2007). *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. Computer Press.

Bruckner, T., Voříšek, J., Buchalcevová, A., Stanovská, I., Chlapek, D., & Řepa, V. (2012). Tvorba informačních systémů. *Praha, Česká republika: Grada Publishing*.

Podpis studenta: Procházka Datum: 10. 11. 2022

Podpis vedoucího práce: Datum: 11. 11. 2022