

Vysoká škola ekonomická v Praze  
Fakulta informatiky a statistiky



**Transformácia business procesov do  
aplikačných procesov**

**DIPLOMOVÁ PRÁCA**

Študijný program: Informační systémy a technologie

Študijný obor: Business analýza

Autor: Bc. Marcel Žec

Vedúci práce: Ing. Oleg Svatoš, Ph.D.

Praha, jún 2023



## **Prehlásenie**

Prehlasujem, že som diplomovú prácu *Transformácia business procesov do aplikačných procesov* vypracoval samostatne za použitia v práci uvedenej literatúry.

V Prahe dňa 28. júna 2023

.....

Podpis študenta



## **Poděkovanie**

Ďakujem vedúcemu práce, ktorým je Ing. Oleg Svatoš, Ph.D., za cenné rady, usmernenia a čas, ktorý mi venoval počas tvorby tejto práce. Poděkovanie patrí aj mojim rodičom za ich neustálu podporu počas celého štúdia na vysokej škole.



## **Abstrakt**

Diplomová práca sa zaoberá problematikou transformácie podnikových procesov na aplikačné procesy pre workflow nástroj Camunda. Práca uvádza čitateľa do podnikového riadenia a popisuje problémy, ktorým si prešlo. Popisuje trendy, ktoré napomáhajú tieto problémy riešiť. Pre dosiahnutie cieľov práce sme najprv analyzovali metodiku MMABP, ktorá sa používa na analýzu a modelovanie podnikových procesov. Ďalej sme analyzovali metodický postup transformácie konceptuálnych modelov do implementačne nezávislých technologických modelov a v neposlednom rade workflow nástroj Camunda. Výsledkom práce je metodický postup transformácie podnikových procesov do aplikačných procesov nástroja Camunda. Ten sa skladá z dvoch nadväzujúcich postupov. Upraveného a doplneného metodického postupu transformácie konceptuálnych modelov vytvorených pomocou metodiky MMABP, do technologických modelov. A metodického postupu transformácie technologických modelov do implementačných modelov nástroja Camunda 8.

## **Klúčové slová**

procesné riadenie, biznis proces, transformácia procesov, workflow, MMABP, Camunda

## **Abstract**

The diploma thesis focuses on the transformation of business processes into application processes using the workflow tool Camunda. The thesis introduces the reader to business management and describes the problems it has gone through. It describes the trends that help solve these problems. To accomplish the goals of the work, we first analyzed the MMABP methodology, which is utilized for analyzing and modeling business processes. Furthermore, we analyzed the methodological procedure for converting conceptual models into implementation-independent technological models. Additionally, we analyzed the Camunda workflow tool. The outcome of this research is a systematic approach to transforming business processes into application processes using the Camunda tool. This approach consists of two distinct procedures. Firstly, an enhanced and refined methodological process is proposed for transforming conceptual models, developed through the utilization of the MMABP methodology, into technological models. Secondly, a methodical procedure is introduced for transforming technological models into implementation models specific to Camunda 8 platform.

## **Keywords**

business process management, business process, process transformation, workflow, MMABP, Camunda



# Obsah

<b>Úvod</b>	<b>21</b>
<b>Metodika</b>	<b>23</b>
<b>Komentovaná rešerš</b>	<b>25</b>
<b>1 Úvod do procesného riadenia a workflow nástrojov</b>	<b>27</b>
1.1 Podnikové procesy a procesné riadenia . . . . .	27
1.2 História a problémy zavedenia procesného riadenia . . . . .	27
1.3 Architektúra orientovaná na služby a procesne riadená architektúra . . . . .	28
<b>2 Vývoj informačného systému</b>	<b>29</b>
2.1 Životný cyklus vývoja . . . . .	29
2.2 Metodika analýzy a modelovania podnikových procesov . . . . .	29
2.3 Analýza a definícia požiadaviek . . . . .	31
2.3.1 Model reality . . . . .	32
2.3.2 Funkčné požiadavky . . . . .	36
2.4 Návrh . . . . .	38
2.4.1 Transformácia konceptuálnych modelov na technologické modely . . . . .	38
2.5 Implementácia . . . . .	39
2.5.1 Procesne riadená aplikácia . . . . .	40
2.5.2 Workflow a workflow management systém . . . . .	41
2.5.3 Implementácia workflow . . . . .	41
<b>3 Camunda</b>	<b>43</b>
3.1 Komponenty a užívateľské zohraniá . . . . .	43
3.1.1 Camunda Modeler . . . . .	43
3.1.2 Tasklist . . . . .	47
3.1.3 Operate . . . . .	47
3.1.4 Optimize . . . . .	47
3.1.5 Zeebe engine a Zeebe klient . . . . .	49
3.1.6 Identity . . . . .	50
3.2 Typy inštalácie a cena . . . . .	50
3.2.1 SaaS Camunda . . . . .	50
3.2.2 Self-managed Camunda . . . . .	51
<b>4 Transformácia konceptuálneho modelu na technologický model</b>	<b>53</b>
4.1 Konceptuálny model . . . . .	53
4.2 Technologický model . . . . .	67
4.2.1 Špecifikácia použitej technológie . . . . .	67
4.2.2 Špecifikácia technických rolí . . . . .	68

4.2.3	Špecifikácia externých udalostí . . . . .	70
4.2.4	Transformácia úloh a pridanie potrebných úloh v závislosti na alternatívnych prechodoch . . . . .	72
4.2.5	Určenie vstupov a výstupov pre každú úlohu procesu . . . . .	77
4.2.6	Aktualizácia diagramu tried . . . . .	80
4.2.7	Špecifikácia notifikácií . . . . .	81
4.3	Zhodnotenie a doplnenie metodického postupu transformácie do technologickej modelu . . . . .	84
4.3.1	Pridanie rozhodovacích modelov a tabuliek . . . . .	85
4.3.2	Špecifikácia business premenných . . . . .	87
4.3.3	Špecifikácia notifikácií . . . . .	90
4.4	Doplnenie technologického modelu . . . . .	91
4.4.1	Špecifikácia business premenných . . . . .	91
4.4.2	Špecifikácia notifikácií . . . . .	99
4.4.3	Pridanie rozhodovacích modelov a tabuliek . . . . .	103
<b>5</b>	<b>Transformácia technologického modelu na implementačný model</b>	<b>105</b>
5.1	Metodický postup transformácie do implementačného modelu nástroja Camunda	106
5.1.1	Vytvorenie štruktúry projektu . . . . .	107
5.1.2	Výber pracovnej jednotky . . . . .	107
5.1.3	Vytvorenie procesného diagramu . . . . .	110
5.1.4	Transformácia procesu . . . . .	114
5.1.5	Tvorba formulárov . . . . .	129
5.1.6	Tvorba rozhodovacích diagramov a tabuliek . . . . .	132
5.1.7	Nasadenie diagramov . . . . .	134
5.1.8	Spustenie diagramov a ich vyladenie . . . . .	134
5.1.9	Problémy a doporučenia . . . . .	136
5.2	Overenie metodického postupu transformácie do implementačného modelu nástroja Camunda . . . . .	142
5.2.1	Vytvorenie štruktúry projektu . . . . .	142
5.2.2	Výber pracovnej jednotky . . . . .	145
5.2.3	Vytvorenie procesného diagramu . . . . .	145
5.2.4	Transformácia procesu . . . . .	146
5.2.5	Tvorba formulárov . . . . .	159
5.2.6	Tvorba rozhodovacích diagramov a tabuliek . . . . .	163
5.2.7	Nasadenie diagramov, spustenie a vyladenie . . . . .	163
5.3	Zhodnotenie nástroja Camunda 8 . . . . .	168
<b>6</b>	<b>Zhrnutie metodického postupu a jeho diskusia</b>	<b>169</b>
6.1	Transformácia konceptuálnych modelov na technologické modely . . . . .	169
6.2	Transformácia technologických modelov na implementačné modely . . . . .	171
6.3	Diskusia . . . . .	172
<b>Záver</b>		<b>175</b>

Zoznam použitej literatúry	177
A Elektronická príloha	183



# Zoznam obrázkov

2.1	Vodopádový model od Royce (Royce 1987, str. 329) . . . . .	30
2.2	Vodopádový model (Sommerwille 2010, str. 30) . . . . .	30
2.3	Model (business) reality (Řepa 2012) . . . . .	32
2.4	Procesná mapa . . . . .	34
2.5	Procesný diagram . . . . .	35
2.6	Diagram tried . . . . .	35
2.7	Stavový diagram . . . . .	36
2.8	Diagram dátových tokov (Řepa 2021, str. 99) . . . . .	37
2.9	Náčrt priebehu procesu v PDA . . . . .	40
3.1	Architektúra platformy Camunda 8 ( <u>Overview Components</u> 2023) . . . . .	43
3.2	Štruktúra projektu v Camunda Web Modeler . . . . .	44
3.3	Tvorba formuláru v Camunda Web Modeler . . . . .	44
3.4	Tvorba BPMN diagramu v Camunda Web Modeler . . . . .	45
3.5	Štruktúra projektu v Camunda Desktop Modeler . . . . .	45
3.6	Tvorba formuláru v Camunda Desktop Modeler . . . . .	46
3.7	Tvorba BPMN diagramu v Camunda Desktop Modeler . . . . .	46
3.8	Camunda Tasklist . . . . .	47
3.9	Camunda Operate . . . . .	48
3.10	Camunda Optimize . . . . .	48
3.11	Camunda Identity ( <u>Identity</u> 2023) . . . . .	50
4.1	Konceptuálny model: procesná mapa . . . . .	55
4.2	Konceptuálny model: diagram klúčového procesu Prenajatie lode č.1 . . . . .	56
4.3	Konceptuálny model: diagram klúčového procesu Prenajatie lode č.2 . . . . .	57
4.4	Konceptuálny model: diagram podporného procesu Nalodenie . . . . .	58
4.5	Konceptuálny model: diagram podporného procesu Vylodenie . . . . .	59
4.6	Konceptuálny model: diagram podporného procesu Riešenie havárie . . . . .	60
4.7	Konceptuálny model: diagram podporného procesu Riešenie škodovej udalosti . . . . .	61
4.8	Konceptuálny model: diagram tried . . . . .	62
4.9	Konceptuálny model: diagram životného cyklu triedy Objednávka . . . . .	63
4.10	Konceptuálny model: diagram životného cyklu triedy Plavba . . . . .	63
4.11	Konceptuálny model: diagram životného cyklu triedy Požiadavka na nalodenie . . . . .	64
4.12	Konceptuálny model: diagram životného cyklu triedy Požiadavka na vylodenie . . . . .	64
4.13	Konceptuálny model: diagram životného cyklu tried Preberací protokol a Odovzdávací protokol . . . . .	64
4.14	Konceptuálny model: diagram životného cyklu triedy Havária . . . . .	65
4.15	Konceptuálny model: diagram životného cyklu triedy Škodová udalosť . . . . .	65
4.16	Konceptuálny model: diagram životného cyklu triedy Loď . . . . .	65

4.17 Konceptuálny model: Diagram dátových tokov kľúčového procesu . . . . .	66
4.18 Konceptuálny model: Diagram dátových tokov podporného procesu Nalodenie . . . . .	66
4.19 Technological specification sketch . . . . .	67
4.20 Technological model: process diagram of key process part 1 . . . . .	73
4.21 Technological model: process diagram of key process part 2 . . . . .	74
4.22 Technological model: process diagram of boarding support process . . . . .	75
4.23 Technological model: class diagram . . . . .	80
4.24 Technological model: process diagram of key process with business variable part 1	97
4.25 Technological model: process diagram of key process with business variable part 2	98
4.26 Technological model: DRD for Cancel decision . . . . .	103
4.27 Technological model: decision table . . . . .	103
 5.1 Vysvetlivky . . . . .	106
5.2 Online Modeler SaaS verzie . . . . .	107
5.3 Dokompozícia implementácia . . . . .	108
5.4 Vytvorenie BPMN diagramu . . . . .	110
5.5 Pridanie plavebných dráh . . . . .	111
5.6 Modelovanie spúšťacích a ukončovacích udalostí . . . . .	112
5.7 Zoznam premenných a konštánt . . . . .	113
5.8 Ukážka transformovaného stavu . . . . .	114
5.9 Transformácia stavov procesu . . . . .	115
5.10 Rozlíšenie udalosti mimo elementu procesného stavu . . . . .	115
5.11 Message Intermediate Catch Event . . . . .	116
5.12 Message End Event . . . . .	116
5.13 Modelovanie úloh a brán . . . . .	117
5.14 Service Task . . . . .	118
5.15 Send Task . . . . .	120
5.16 Business Rule Task . . . . .	121
5.17 Script Task . . . . .	122
5.18 Call Activity 1 . . . . .	123
5.19 Call Activity 2 . . . . .	124
5.20 User Task - chybajúci formulár . . . . .	124
5.21 Exclusive gateway (XOR) / Inclusive gateway (OR) . . . . .	125
5.22 Event based gateway . . . . .	125
5.23 Vytvorenie objektu . . . . .	126
5.24 Zmena stavu objektu . . . . .	127
5.25 Ukladanie stavu objektu . . . . .	128
5.26 Tvorba formulárov . . . . .	129
5.27 Napojenie formulára na proces v offline modeléry . . . . .	130
5.28 Napojenie formulára na proces v online modeléry . . . . .	131
5.29 Mapovanie formulárových vstupov na výstupy úlohy . . . . .	131
5.30 Vytvorenie súboru pre rozhodovací diagram . . . . .	132
5.31 Implementačný DRD diagram . . . . .	132

5.32 Naplnenie rozhodovacej tabuľky . . . . .	133
5.33 Nasadenie diagramov . . . . .	134
5.34 Kontrola funkčnosti cez rozhranie Operate . . . . .	135
5.35 Zacyklenie dvoch časových udalostí . . . . .	136
5.36 Riešenie zacyklenia dvoch časových udalostí . . . . .	137
5.37 Nastavenie stavu viacerým objektom pomocou jednej úlohy . . . . .	138
5.38 Problém s udalosťou v rozhraní Tasklist . . . . .	139
5.39 Úprava procesného stavu pri vynechaní udalostí . . . . .	140
5.40 Testovanie s dlho trvajúcimi časovými odalostami . . . . .	141
5.41 Implementation: Web interface . . . . .	143
5.42 Implementation: Web interface for order creation . . . . .	143
5.43 Implementation: Inputs mapping in start event . . . . .	145
5.44 Implementation: variables and constants in CONSTANTS script task . . . . .	146
5.45 Implementation: first work unit . . . . .	146
5.46 Implementation: message event configuration . . . . .	147
5.47 Implementation: Email log . . . . .	148
5.48 Implementation: Email configuration . . . . .	148
5.49 Implementation: timer event cycle . . . . .	150
5.50 Implementation: first work unit . . . . .	150
5.51 Implementation: work unit with service task, script task and support process call	151
5.52 Implementation: Service task . . . . .	152
5.53 Implementation: Script task creating object . . . . .	152
5.54 Implementation: Call activity starting support process . . . . .	152
5.55 Implementation: Messages from support process . . . . .	153
5.56 Implementation: Saving state . . . . .	154
5.57 Implementation model: process diagram of key process part 1 . . . . .	155
5.58 Implementation model: process diagram of key process part 2 . . . . .	156
5.59 Implementation model: process diagram of key process part 3 . . . . .	157
5.60 Implementation model: process diagram of support process Boarding . . . . .	158
5.61 Implementation: Form for task Contact customer . . . . .	159
5.62 Implementation: Form for task Check all documents . . . . .	159
5.63 Implementation: Form for task Hand over boat . . . . .	160
5.64 Implementation: Form for task Check reported problem . . . . .	160
5.65 Implementation: Form for task Fixing the problem . . . . .	161
5.66 Implementation: Form for task Request for control end . . . . .	161
5.67 Implementation: Form for task Contact customer on cruise . . . . .	162
5.68 Implementation: Form for task Submit for signature . . . . .	163
5.69 Implementation: Inputs mapping and DRD diagram . . . . .	164
5.70 Implementation: Decision table . . . . .	165
5.71 Implementation: Testing a debugging in Operate 1 . . . . .	166
5.72 Implementation: Testing a debugging in Operate 2 . . . . .	167
6.1 TeamAssistant model (Tauchmanová 2023, str. 100) . . . . .	172

# Zoznam tabuliek

4.1	List of technical roles . . . . .	68
4.2	List of external events from key process . . . . .	70
4.3	List of external events from support process Boarding . . . . .	71
4.4	List of inputs and output for tasks in key process . . . . .	78
4.5	List of inputs and output for tasks in support process Boarding . . . . .	79
4.6	Notification for deposit payment . . . . .	81
4.7	Notification for full price payment . . . . .	81
4.8	Notification for cancel reservation . . . . .	81
4.9	Notification for cancel reservation without refund . . . . .	82
4.10	Notification for completed order . . . . .	82
4.11	Notification for changed boarding date . . . . .	82
4.12	Notification for cancel order with penalty . . . . .	83
4.13	Notification for exceptionally canceled order . . . . .	83
4.14	Notification for finished order . . . . .	83
4.15	Zoznam business premenných . . . . .	89
4.16	Rozšírená tabuľka notifikácií s vyznačenými špeciálnymi premennými . . . . .	91
4.17	List of business variables . . . . .	96
4.18	Notification for deposit payment . . . . .	99
4.19	Notification for full price payment . . . . .	99
4.20	Notification for cancel reservation . . . . .	99
4.21	Notification for cancel reservation without refund . . . . .	100
4.22	Notification for completed order . . . . .	100
4.23	Notification for changed boarding date . . . . .	100
4.24	Notification for cancel order with penalty . . . . .	101
4.25	Notification for exceptionally canceled order . . . . .	101
4.26	Notification for finished order . . . . .	102

# Zoznam výpisov kódu

3.1	Zeebe klient pre NodeJS . . . . .	49
5.1	Debugging skripts . . . . .	144
5.2	Web application . . . . .	144
5.3	Email templates . . . . .	149
5.4	Job workers . . . . .	151
5.5	Function documentation . . . . .	151
5.6	Event worker . . . . .	153
5.7	Saving files . . . . .	154



# Zoznam použitých skratiek

**WfMS** Workflow Management Systém

**MMABP** Metodika modelování a analýzy podnikových procesů

**SOA** Service Orientated Architecture - architektúra orientovaná na služby

**PDA** Proces Driven Application - procesne riadená aplikácia

**BPMN** Business Process Model and Notation

**DMN** Decision Model and Notation

**DRG** Decision Requirements Graph - graf rozhodovacích požiadaviek

**DRD** Decision Requirements Diagram - diagram rozhodovacích požiadaviek

**TAS** TeamAssistant



# Úvod

Posledné roky sme boli svedkami globalizácie a digitalizácie. Globalizácia vniesla do podnikania väčšiu mieru súťaživosti. Digitalizácie zasa do veľkej miery prepojila podniky s informačnými technológiami. Začali vznikať podniky, ktoré bez informačných technológií nemôžu fungovať, pretože je na nich postavený ich podnikateľský plán. V takomto silne digitálnom a konkurenčnom prostredí musia byť podniky ale aj ich informačné systémy flexibilné, aby sa mohli rýchlo prispôsobiť novým požiadavkam trhu.

Tu podnikom pomáha procesné riadenie. Hlavnou hnacou silou a jedným z cieľov procesného riadenia je snaha dosiahnuť flexibilitu v podniku na organizačnej ale aj operačnej úrovni. Ďalším dôležitým cieľom procesného riadenia je zmenšovanie prieplasti medzi business procesmi a informačných technológií, ktoré podniky používajú (Weske 2019). Táto prieplast spôsobovala problémy v deväťdesiatych rokoch. Vela firiem sa snažilo o reorganizáciu a prechod na procesné riadenie (business process reengineering), ale pri tejto zmene zlyhalo. Existujúce informačné systémy sa nedokázali prispôsobiť novej štruktúre podniku a novému spôsobu riadenia (Sneed a Verhoef 2001).

K zvýšeniu flexibility informačných systémov napomohlo napríklad agilné riadenie softvérových projektov. Veľmi významným posunom vpred bola predovšetkým softvérová servisne orientovaná architektúra (Stiehl 2014), či veľmi populárny architektonický štýl microservíš (Dragoni et al. 2017).

Ďalším aspektom, ktorý je dnes pre podniky relevantný je automatizácia. Niektoré činnosti business procesu alebo aj celé procesy môžu byť čiastočne alebo plne automatizované. Iné zasa vyžadujú interakciu s človekom, napríklad zamestnancom podniku, ktorý má nejakú konkrétnu rolu. Súhrne tieto procesné činnosti, prípadne celé procesy v kontexte informačných systémov označujeme ako **workflow** a pomocou nástrojov, ktoré označujeme ako **workflow management systém (WFMS)**, ich môžeme namodelovať, nastavovať rôzne pravidlá či obmedzenia. Následne vniká aplikácia, v ktorej prebieha životný cyklus podnikového proces (Weske 2019). WFMS obvykle ponúka niekoľko druhov užívateľských rozhraní. Bežný užívatelia v nich môžu vykonávať procesné aktivity, ktoré sú im priradené. Administrátorské prostredie umožňuje napríklad nastavovať role užívateľov. Dôležitou vlastou administrátorského rozhrania je možnosť sledovať celý životný cyklus procesu. To prináša niekoľko výhod ako možnosť analyzovať životný cyklus procesu a následne ho optimalizovať.

Ako sa ale podnik dostane k tomu, že má takúto aplikáciu alebo systém, ktorý pozná podnikový proces spoločnosti?

Najprv je potrebné zanalyzovať business proces podniku a správne ho namodelovať. Pri analýze je potrebné používať existujúce metodiky a postupy. Dobrým príkladom takejto metodiky je MMABP (Řepa 2021), ktorá vznikla na Fakulte informatiky a štatistiky

Vysokej školy ekonomickej v Prahe a jej autorom je profesor Řepa.

Pri modelovaní podnikových procesoch, ktoré majú byť podporované informačnými technológiami vznikne niekoľko typov modelov podľa vrstiev architektúr. Z prvej konceptuálnej vrsty prebieha transformácia do implementačne nezávislej technologickej vrstvy. Pri tejto transformácii môžeme použiť metodický postup, ktorý vytvoril Pavelčák (2021) a upravila ho Tauchmanová (2023). Poslednou vrstvou, ktorá umožňuje spustenie aplikácie je implementačná. Implementačné modely sú oproti ostatným modelom závislé na konkrétnom workflow nástroji a pracujú s nimi viac vývojári ako analytici. Preto sa líšia od ostatných modelov na ostatných vrstvách. Každý workflow nástroj ponúka inú škálu dostupných predprogramovaných funkcionálit a môže mať špecifické požiadavky na vstupy alebo vyžaduje špecifické úkony pri jeho používaní. Informácie o funkcionalite a špecifikách rôznych workflow nástrojov vieme obvykle nájsť v dokumentáciách a návodoch pre dané nástroje. Problém ale nastáva pri prechode z platformovo nezávislých procesných modelov na platformovo závislé implementačné modely. Práve na tento problém sa v diplomovej práci zameriame. Vytvoríme metodický postup pre transformáciu business procesov do aplikačných procesov workflow nástroja s názvom Camunda. Celý postup sa bude skladať z krokov už existujúceho metodického postupu transformácie do technologických modelov (Pavelčák, Tauchmanová) a novo vzniknutého postupu transformácie do implementačných modelov nástroja Camunda.

Ciele diplomovej práce sú:

- Overiť metodický postup transformácie konceptuálnych modelov do technologických modelov na konkrétnom business procese
- V prípade nedostatkov navrhnuť úpravy alebo doplniť metodický postup transformácie konceptuálnych modelov do technologických modelov
- Navrhnuť metodický postup transformácie technologických modelov do implementačných modelov workflow nástroja Camunda
- Aplikovať metodický postup transformácie technologických modelov do implementačných modelov workflow nástroja Camunda na konkrétnom business procese

Metodický postup, ktorý bude výsledkom práce môže byť použitý ako návod pri implementovaní podnikových procesov pomocou nástroja Camunda. Ďalšie uplatnenie môže nájsť pri porovnanie implementačnej náročnosti rôznych workflow nástrojov.

# Metodika

Výzkumnou metódou tejto diplomovej práce je **Desing science research**. Cieľom tejto metódy je vytvoriť artefakt, ktorý je možné aplikovať na riešenie nejakého konkrétneho organizačného problému vo vybranej doméne (Hevner et al. 2004).

Prvou dôležitou súčasťou každej diplomovej práce je rešerš literatúry. Pre pochopenie výskumného problému, je potrebné oboznámiť sa s doménou problému. V tejto práci ide o problematiku transformácie procesných modelov, ktorá je veľmi úzko spojená s procesným riadením a jeho analýzou. Na vysvetlenie základných pojmov a uvedenie do kontextu problému je potrebné čerpať z odbornej literatúry. Zároveň nám rešerš umožní získať prehľad o aktuálnom stave danej problematiky. Rešerš tejto práce je organizovaná kombinované tématicko-chronologicky.

Výsledkom práce bude artefakt v podobe metodického postupu. Tento postup bude popisovať konkrétnie kroky, ktoré treba vykonať pri transformovaní business procesov do aplikačných procesov v konkrétnom workflow nástroji.

Na splnenie cieľov diplomovej práce sme zvolili nasledujúci postup. Začíname analyzovaním metodiky MMABP a jej princípov. Hlavným výstup analýzy musí byť jasné pochopenie rozdielov medzi jednotlivými architektúrami procesu a pochopenie postupov analyzovania a modelovania konceptuálneho modelu.

Následne budeme analyzovať prechody medzi týmito architektúrami a ich modelmi. Preskúmame metodický postup Pavelčáka (2021) a Tauchmanovej (2023) na transformáciu konceptuálnych modelov do technologických modelov a overíme ho na konkrétnom business procese.

Ďalej budeme analyzovať nástroj Camunda. Budeme analyzovať predovšetkým jeho dokumentáciu ([Camunda Platform 8 Docs 2023](#)) a knihu Real-Life BPMN (2019). Identifikujeme všetky potrebné vstupy pre tvorbu implementačného modelu v tomto nástroji a jeho špecifiká.

Porovnaním preskúmaného metodického postupu a identifikovanými potrebami nástroja Camunda overíme, či nie je potrebné metodický postup doplniť alebo upraviť.

Z predošlých analýz a prípadných úprav existujúcich postupov navrhнемe metodický postup prechodu z technického modelu do implementačného modelu nástroja Camunda.

Na rovnakom business procese, použitom pri overovaní transformácie do technologických modelov, overíme navrhnutý metodický postup transformácie do implementačných modelov nástroja Camunda.

Na záver zhrnieme celý metodický postup transformácie business procesov do aplikačných procesov nástroja Camunda.



# Komentovaná rešerš

V tejto práci je dôležité uviesť čitateľa do problematiky procesného riadenia, jeho cieľov a prínosov pre podniky. Dobrý zdrojom je kniha od autora Weske (2019). Ponúka jasné definície najdôležitejších pojmov v oblasti procesného riadenia ale aj širší pohľad na potrebu vzájomnej prepojenosti medzi business cielmi a používanými technológiami v podniku. Ponúka rýchle zasvätenie do notačného jazyk BPMN ale aj do pojmov súvisiacich s workflow nástrojmi, ktoré sú pre túto prácu veľmi dôležité.

Čitateľovi je dôležité ukázať, akými problémami si prešlo procesné riadenie. Budeme čerpať od autora Sneed 2001 informácie o problémoch pri business process reengineeringu v deväťdesiatych rokoch. Tým chceme priblížiť a zvýrazniť potrebu a prínosy workflow nástrojov.

Presunieme sa do súčasnosti a popíšeme si, ako sa technológie zmenili. Autora Stiehl (2014) popisuje nové trendy v oblasti projektové riadenia ale aj softvérových architektúr. Zároveň popisuje nedostatky týchto zmien, ktoré nedokázali zlepšiť flexibilitu. Spomenieme aj najnovšie trendy v oblasti softvérových architektúr. Dragoni (2017) nám ponúka definíciu microservisnej architektúry ale aj popis toho, čo je potrebné na jej dobré manažovanie.

Pre správne pochopenie kontextu celej práce je potrebné popísat aj životný cyklus vývoja informačných systémov. Tu sa budeme opierať o (Sommerwille 2010) ale poukážeme aj na staršie interpretácie životných cyklov vývoja (Benington 1987), (Royce 1987).

V práci sa zaoberáme transformáciou podnikových procesov, čoho súčasťou je analýza. Pri analýze je potrebné držať sa metodík a metodických postupov. Budeme sa zaoberať metodiku MMABP. Princípy a postupy metodiky nájdeme v niekoľkých knihách od profesora Řepy. Od staršej publikácie z roku 2007 až po publikáciu z roku 2021.

Existuje niekoľko akademických prác, v ktorých sa používa nástroj Camunda. Napríklad Alexey (2017) alebo Lipowski (2022). Tieto ale aj ďalšie akademické práce popisujú analýzu a následný vývoj pre konkrétny business problém. Môžeme z nich čerpať informácie o problémoch či konkrétnych špecifikkach Camundy. Existujú však dve akademické práce z ktorých budeme používať metodický postup transformácie konceptuálnych modelov do technologických modelov. Ide o diplomovú prácu pôvodného autora metodického postupu Pavelčáka (2021). A diplomovú prácu Tauchmanovej (2023), ktorá overila a upravila metodický postup.

Pri vytváraní metodického postupu transformácie do implementačných modelov nástroja Camunda budeme primárne vychádzať z online dokumentácie nástroja ([Camunda Platform 8 Docs 2023](#)) ale aj z knihy Real-Life BPMN (Freund a Rücker 2019). Použijeme aj existujúce znalosti transformácie procesných stavov v nástroji Camunda (Řepa 2022).



# 1. Úvod do procesného riadenia a workflow nástrojov

## 1.1 Podnikové procesy a procesné riadenia

Pozrime sa nato, ako podnikové procesy a procesné riadenie definuje Weske (2019). Podnikový proces je spojením aktivít, ktoré sa vykonávajú v podnikoch za účelom dosahovať ciele podniku. Ide o aktivity na rôznych podnikových oddelenia, či už priamo vo výrobe ale aj v organizačných oddeleniach podniku. Podnikové procesy jedného podniku môžu byť prepojené s podnikovým procesom iného podniku. Tradične sa v podnikoch vykonáva podnikový proces manuálne pracovníkmi. Postup aktivít podnikového procesu je definovaný v nejakom podnikovom predpise, ktorým sa zamestnanci riadia. Skôr ako sa presunieme ku modernejšiemu poňatiu podnikových procesov si ešte musíme definovať procesné riadenia. Procesným riadením súhrne označujeme činnosti, pomocou ktorých sa navrhujú, analyzujú a spravujú podnikové procesy.

## 1.2 História a problémy zavedenia procesného riadenia

S rozvojom informačných technológií začali byť informačné systémy podnikov plne podporované informačnými a komunikačnými technológiami. V deväťdesiatych rokoch začali podniky prechádzať na procesné riadenie, o ktorom pojednáva Hammer a Champy (1993). Táto radikálna zmena dopadla vo veľa podnikoch veľmi zle a podniky sa vrátili ku predošlému spôsobu riadenia. Sneed (2001) v svojej publikácii zhrnul problémy, ktoré viedli k neúspechu pri prechode na procesné riadenie. Technológie neboli schopné udržať krok, s novým spôsobom riadenia podniku. Informačné systémy veľkého množstva spoločností používali technológie zo sedemdesiatych rokov a šlo o monolitické systémy. Monolitické systému sú obvykle príliš komplexné, náročné na údržbu a nový vývoj. Ich úprava v takom rozsahu bola teda prakticky nerealizovateľná.

Technológie sa ďalej vyvíjali a trhy začali byť veľmi dynamické. Podniky začali byť nútene ponúkať lepšie a špecifickejšie produkty pre svojich zákazníkov. Aby podniky v dnešnej dynamickej dobe dokázali držať krok s konkurenciou, je pre nich veľmi dôležitou vlastnosťou flexibilita a adaptivita. A to nie len na organizačnej úrovni podniku, ale aj na úrovni ich informačných systémov. Tieto dva podnikové zložky majú medzi sebou obvykle veľkú pripasť, ktorá neumožňuje dosahovanie spomínaných vlastností. Preto je dôležité aby sa táto pripasť zužovala. (Weske 2019)

### **1.3 Architektúra orientovaná na služby a procesne riadená architektúra**

Jednou z klúčových softvérových architektúr, ktorá oproti staršej monolitickej architektúre ponúka výrazne väčšiu flexibilitu je **architektúra orientovaná na služby (SOA)**. Táto architektúra delí systém na menšie časti, ktoré nazývame služby alebo servisy. Hlavnou výhodou má byť znova-použiteľnosť služieb. SOA ale nemá presnú definíciu a pri zlom návrhu môže dochádzať k silným väzbám medzi službami, čím sa vytrácajú výhody tejto architektúry (Stiehl 2014).

Ďalším evolučným krokom SOA bol vznik architektonického štýlu microservis (Dragoni et al. 2017), ktorý ponúka veľa dobrých vlastností ako je flexibilita, udržateľnosť či škálovateľnosť. Na druhú stranu so sebou prináša aj negatíva, ktoré sa často prehliadajú ako napríklad sietová komplexita celého systému či komplikovanejšie zabezpečenie systému, a v neposlednom rade aj vysoká počiatočná investícia pri prechode na tento architektonický štýl.

V predošlých sekciách sme nikde nespomenuli podnikové procesy. Každá softvérová architektúra ponúka isté výhody ale aj nevýhody. Pri procesnom riadení je dôležité aby sa vedel informačný systém prispôsobovať podnikovým procesom. Preto nie je vhodné aby sme sa na architektúru softvéru a podnikové procesy pozerali oddelené. Preto je dobré zaviesť pojem **procesne riadená architektúra**, ktorý podľa Stiehl (2014) môžeme zjednodušíť ako prepojenie procesného riadenia s architektúrou orientovanou na služby. V skutočnosti pri procesne riadenej architektúre nemusí byť použitá SOA. Obe architektúry ale kladú dôraz na znova-použiteľnosť. Preto ak ma podnik svoje systémy postavené na SOA alebo microservisnom architektonickom štýle, prechod na procesne riadenú architektúru bude výrazne jednoduchší. Presuňme sa k tomu, ako prebieha vývoj informačných systémov v kontexte procesného riadenia a akú podobu má informačný systém postavený na procesne riadenej architektúre.

## 2. Vývoj informačného systému

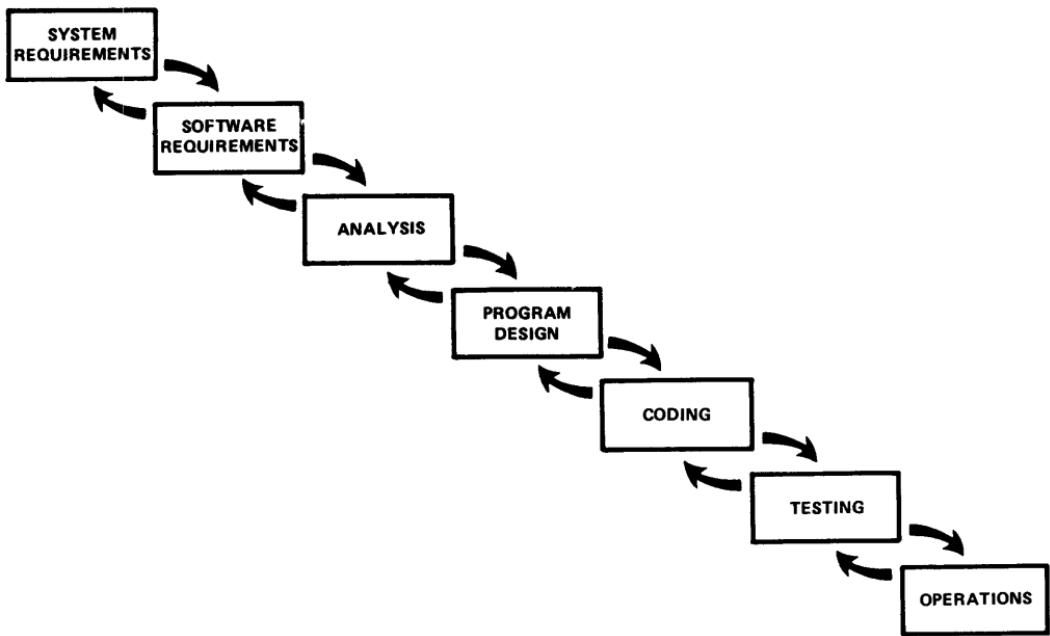
V tejto kapitole sa pozrieme na životný cyklus vývoja informačného systému. Je totiž veľkým prínosom ak všetci aktéri, ktorí sa na vývoji podieľajú, od business analytikov cez programátor až po testerov, vnímajú existenciu jednotlivých fáz a uvedomujú si komplexitu takého vývoja. Popíšeme si princípy metodiky MMABP, ktorá sa používa pri vývoji informačných systémov pre procesne riadené organizácie. Metodika sa sice primárne zaoberá analytickými fázami celkového vývoja ale jej princípy sú uplatnitelné aj v ďalších fázach, ktorími sa v práci zaoberáme. Popíšeme si teda princípy metodiky a konkrétnu postupy zasadíme do kontextu životného cyklu vývoja. Vo fázach, ktoré MMABP nepokrýva si predstavíme metodické postupy, ktoré je vhodné nasledovať. Zároveň si zavedieme a vysvetlíme zopár pojmov, ktoré je nevyhnutné pri vývoji informačného systému pre procesne riadený podnik poznat.

### 2.1 Životný cyklus vývoja

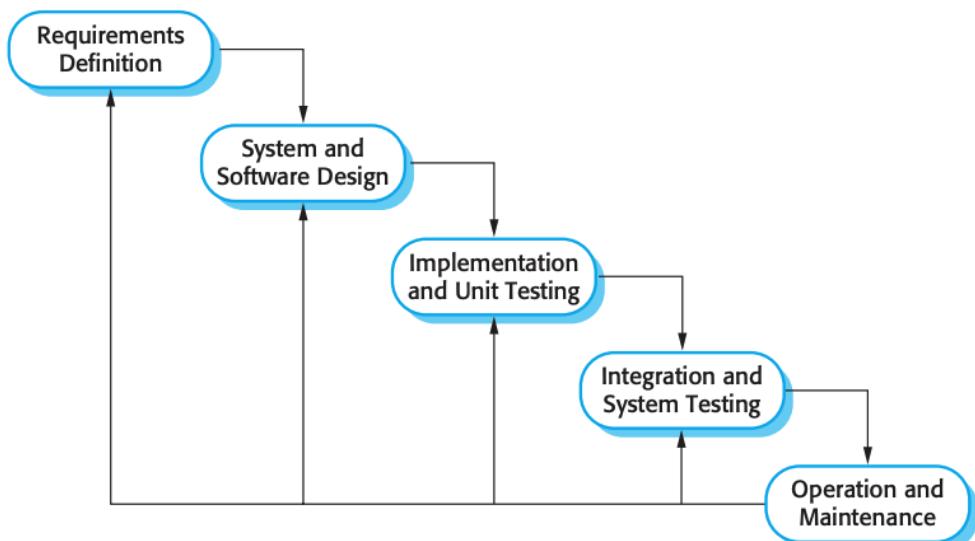
Existujú rôzne modely životných cyklov vývoja, ktoré sú závislé na metodike, podľa ktorej vývoj prebieha. Najstarším modelom životného cyklu, ktorý zdokumentoval Benington v roku 1956, je v vodopádový model. Neskôr ho v roku 1970 upravil Royce (obrázok č.2.1) a v súčasnosti sa obvykle používa interpretácia, ktorú vidíme na obrázku 2.2. V zásade je to stále Royce-ov model akurát sa niektoré kroky pospájali prípadne premenovali. V dnešnej dobe sú veľmi populárne aj agilné metodiky vývoja (Martin c2014) softvéru. V princípe sa ale aj v týchto metodikách opakujú kroky, ktoré nájdeme v Royce-ovom modeli, akurát nemusia byť explicitne vyjadrené. Preto sa agilnými metodikami nebudem podrobnejšie zaoberať.

### 2.2 Metodika analýzy a modelovania podnikových procesov

Ako nám už názov napovedá, metodika známa aj pod skratkou MMABP definuje princípy a techniky na analyzovanie podnikových procesov a ich modelovanie. Vznikla na katedre informačných technológií Vysokej školy ekonomickej. Skôr ako sa presunieme ku technikám a postupom analýzy a modelovanie, je potrebné si predstaviť tri základne a vzájomne súvisiace princípy metodiky MMABP.



Obr. 2.1: Vodopádový model od Royce (Royce 1987, str. 329)



Obr. 2.2: Vodopádový model (Sommerwille 2010, str. 30)

### **Princíp modelovania**

Model je formálnym vyjadrením a zjednodušeným zobrazením skúmaného javu, v našom prípade informačného systému. Princíp modelovanie vyjadruje predpoklad, že informačný systém je modelom reálneho sveta, teda objektívnym základom k implementácií musia byť reálne skutočnosti, ktoré existujú mimo a nezávisle na organizácii.

### **Princíp abstrakcie**

Tento princíp vyjadruje spôsob, akým môžeme identifikované skutočnosti skúmaného problému, pomocou hierarchickej abstrakcie rozdeliť do menších celkov. A následne analyzovať tieto celky samostatne do detailnejšej úrovne pohľadu. Existujú dva základné typy hierarchickej abstrakcie:

- **Agregácia** - hierarchicky nižší prvok je časť vyššieho prvku (napr. proces sa skladá z činnosti, ktoré sú súčasťou procesu)
- **Generalizácia** - hierarchicky nižší prvok je špecifickou variantou vyššieho prvku (napr. entity *zákazník* a *zamestnanec* môžu byť len špecifikáciou entity *človek*)

### **Princíp rôznych architektúr procesu**

Tento princíp vyjadruje potrebu oddeliť rôzne typy charakteristík procesu do vrstiev. Definuje teda spôsob, akým používame abstrakciu pri vývoji informačného systému. Architektúry procesu rozdeľujeme nasledovne do troch vrstiev:

- **Konceptuálna** - určuje **čo** je obsahom systému, nezahrňuje technickú koncepciu ani implementačné detaily
- **Technologická** - určuje **ako** bude obsah systému realizovaný pomocou danej technológie, zahrňuje technickú koncepciu riešenie (organizácia a spracovanie dát)
- **Implementačná** - určuje **čím** je technologické riešenie realizované, zohľadňuje implementačné špecifické použitého vývojového prostredia (databázy, programovacieho jazyka, komunikačného protokolu a pod.)

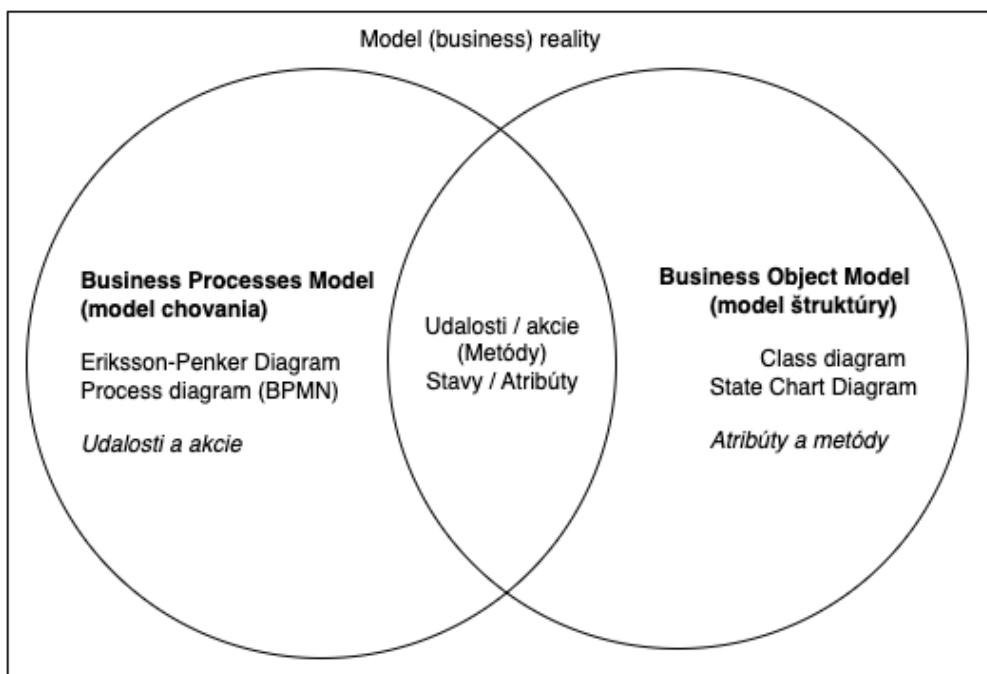
## **2.3 Analýza a definícia požiadaviek**

Prvé tri kroky Royce-ového modelu sú systémové požiadavky, funkčné požiadavky a analýza (obrázok 2.1). Modernejšia interpretácia tieto kroky spája do jedného kroku označovaného ako analýza a definícia požiadaviek, prípadne iba ako analýza. Pomocou metodiky MMABP sme schopní vykonať tieto analytické kroky pomocou modelov, ktoré vytvárame na prej, teda **konceptuálnej vrstve** procesnej architektúry. Fázu systémových požiadaviek a fázu

analýzy pokrýva v MMABP tzv. **model reality**. Softvérové požiadavky a ich definícia nám potom vzniká na základe **diagramu dátových tokov**.

### 2.3.1 Model reality

Prvým modelom, ktorý v analýze vytvárame je model reality, ktorý sa skladá z objektového modelu a procesného modelu (obrázok č.2.3). **Objektový model** sleduje základné stavebné kamene, z ktorých sa realita skladá a popisuje tak jej štruktúru. Definuje teda objekty, ich vlastnosti a vzájomné vzťahy. **Procesný model** popisuje chovanie reality, a teda definuje časovo nadväzné akcie v závislosti na udalostiach a stavoch procesu.



Obr. 2.3: Model (business) reality (Řepa 2012)

Pri modelovaní reality potrebujeme dosiahnuť vhodný kompromis medzi úplnosťou a podrobnosťou, pretože obvykle je zložitosť reálneho systému veľmi veľká. Na dosiahnutie tohto kompromisu sa na problém môžeme pozerať z globálneho hľadiska a z detailného hľadiska. **Globálny pohľad** má za úlohu zachytiť úplnosť systému na úkor veľkých detailov. **Detailný pohľad** zachycuje veľkú mieru detailov nejakej menšej časti systému. Kombináciou dvoch modelov a dvoch pohľadom dostávame štyri rôzne modely.

#### Globálny model systému procesov

Popisuje existenciu procesov a vzťahy medzi nimi. Tvorí ho tzv. **procesná mapa**, ktorú môžeme vytvoriť pomocou Eriksson-Penker notácie (Eriksson a Penker c2000) alebo pomocou modernejšieho TOGAF Event diagramu (T. O. Group 2018). V metodike MMABP rozlišujeme, a teda aj v procesnej mape sa nachádzajú dva druhy procesov (Řepa

a Svatoš 2021) a to kľúčové a podporné procesy. **Kľúčový proces** priamo slúži na uspokojenie potrieb zákazníka. **Podporný proces** svojim výstupom napomáha a podporuje ostatné procesy (kľúčové ale aj podporné). Procesy, ktoré spolu tvoria jeden celok, vyjadrujúci nejakú schopnosť podniku nazývame **business funkciou**. Ukážku procesnej mapy vidíme na obrázku č.2.4.

### **Detailný model procesu**

Detailne popisuje priebeh procesu ako usporiadanie štruktúru akcií potrebných k dosiahnutiu cieľa procesu v kombinácií so stavmi procesu (Řepa 2012). Vychádza z procesov v globálnom modeli, ktoré popisuje v rozumnom detaile (Řepa a Svatoš 2021). Vytvára sa pomocou štandardného jazyka BPMN (O. M. Group 2011). Skladá sa z niekoľko hlavných elementov:

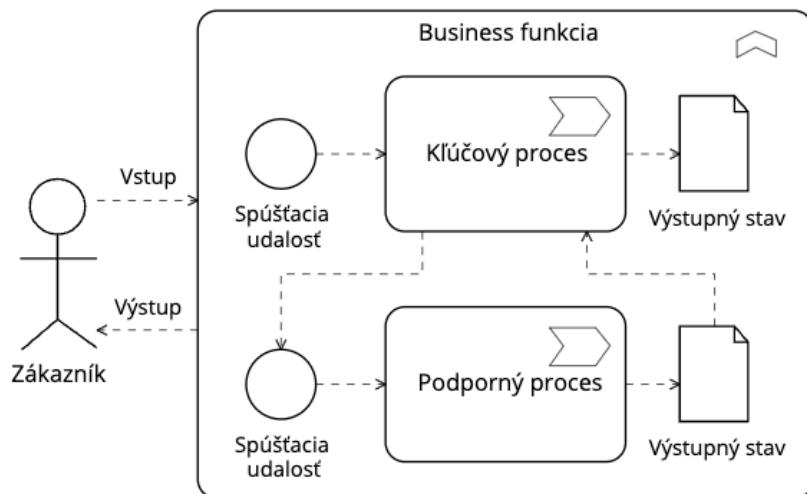
- **Stimul (podnet)** - je podnetom pre nejakú činnosť a obvykle je vyvolaný udalosťou alebo riadiacou činnosťou
- **Stav** - každá objektívne nutná prestávka medzi dvoma činnosťami pri čakaní na nejaký vonkajší (mimo proces) stimul
- **Činnosť** - elementárna alebo komplexná činnosť

Detail procesu môžeme vnímať z dvoch úrovní. Prvým vonkajším hľadiskom je tzv. **procesný krok**. Ukazuje miesta, kde sa proces zastaví (stav procesu) a následne čaká na nejakú spätnú väzbu (udalosť). Popisuje tak synchronizáciu procesu so svojim okolitým prostredím. Procesným krokom môže byť elementárna činnosť ale často to býva nejaká komplexná činnosť. Aby sme vedeli zachytiť aj vnútorný detail procesného kroku, teda popísat řetazec jednotlivých úloh tak vytvárame aj diagram na úrovni úloh. **Úlohu** treba chápať ako pracovnú jednotku, ktorá mení stav nejakého, pre business významného objektu a je najväčším detailom, ktorý v tomto modeli vyjadrujeme (Řepa a Svatoš 2021). Ukážku procesného diagramu spolu s jedným detailným pohľadom na procesný krok vidíme na obrázku č.2.5.

### **Globálny model systému objektov**

Popisuje systém objektov a ich vzájomných vzťahov. Vytvára sa pomocou diagramu tried jazyka UML. Kedže sa nachádzame na konceptuálnej úrovni, tak tento diagram nemodelujeme databázovým či aplikačným prístupom. Nejde o technologický návrh objektov ale o akýsi obchodný slovník. Chceme formálne zachytiť spôsob, akým podnik chápe realitu. Trieda objektu je množina rovnakého druhu objektov. Teda objektov s rovnakými vlastnosťami a operáciami. V diagrame ďalej vyznačujeme sémantické vzťahy medzi triedami (asociácie) a ich kvantifikáciu (kardinalitu) a kvalifikáciu (parcialitu). V neposlednom rade tu patrí aj dedičnosť tried (generalizácia), ktorá popisuje dedenie atribútov a operácií potomkami (Řepa 2012)(Řepa a Svatoš 2021). Ukážku diagramu tried

vidíme na obrázku č.2.6.

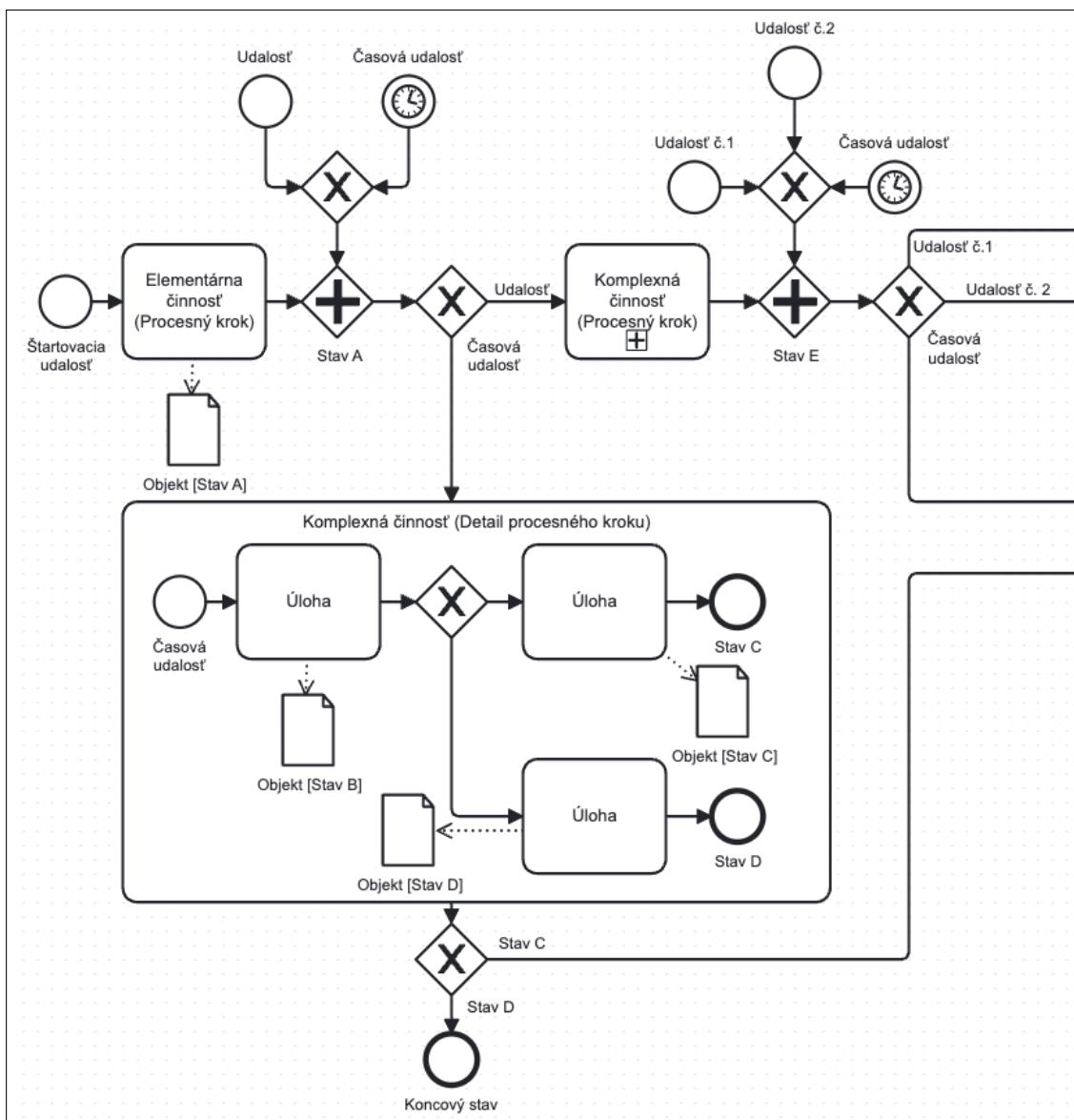


Obr. 2.4: Procesná mapa

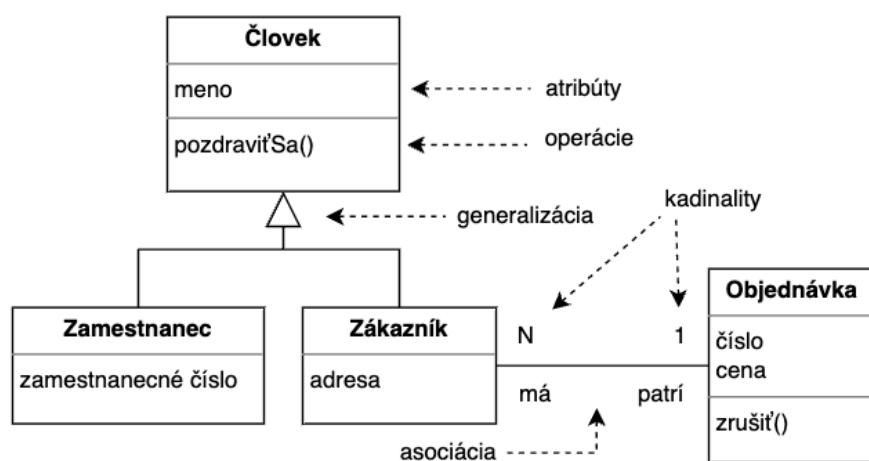
### Detailný model objektov

Tento model popisuje životný cyklus objektov (zovšeobecnený na triedy objektov), teda vytvára stavy objektov a pravidlá prechodu medzi týmito stavmi. Nie je vhodné vytvárať životný cyklus všetkých tried. Potrebujeme zachytiť stavy významných tried, teda takých ktorých stav je výsledkom nejakej úlohy a následne ovplyvňujú ďalšie činnosti procesu.

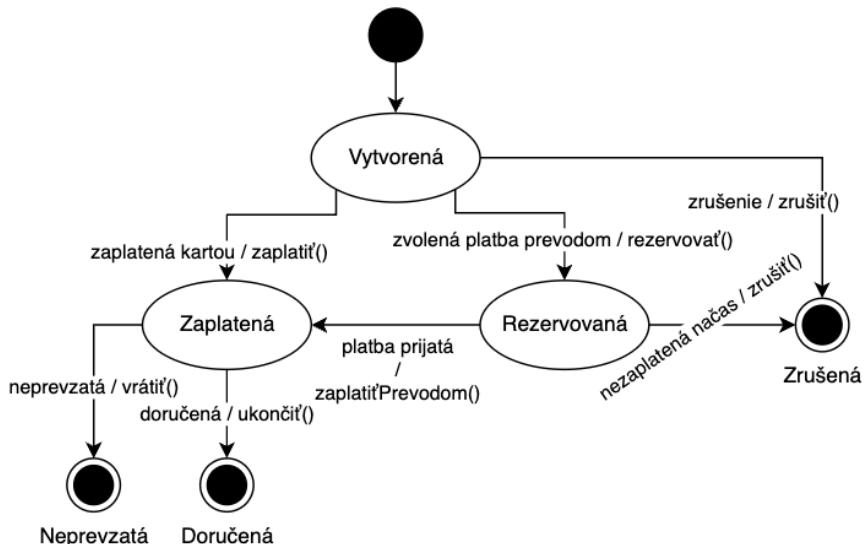
Takisto treba bráť na zreteľ, že stav je akýmsi významným mŕtlikom pre business a daný objekt (Řepa a Svatoš 2021). Niekedy totiž intuitívne vytvárame oveľa viac stavov, ktoré túto požiadavku nespĺňajú a sú tak pre business a konceptuálny model zbytočné. Model vytvárame pomocou stavového diagramu jazyka UML (O. M. Group 2017). Medzi jednotlivými stavmi popisujeme udalosť alebo dôvod na uskutočnenie prechodu a následne operáciu danej triedy, ktorá je zodpovedná za prechod (Řepa 2012). Na obrázku č.2.7 vidíme ukážku stavového diagramu triedy *objednávka*.



Obr. 2.5: Procesný diagram



Obr. 2.6: Diagram tried



Obr. 2.7: Stavový diagram

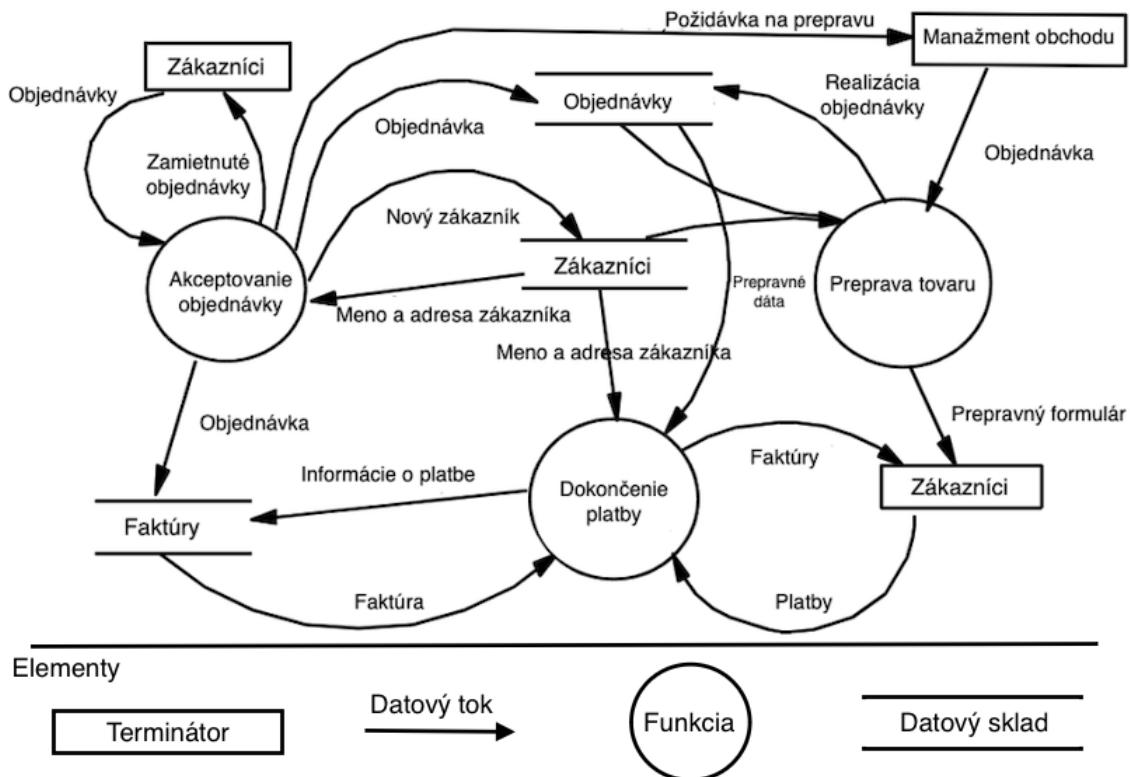
### 2.3.2 Funkčné požiadavky

Pri popise detailného modelu procesu sme spomínali, že úloha mení stav objektu a je najvyšším detailom, ktorý modelujeme. Úloha ale môže v skutočnosti pozostávať z viacerých operácií. Napríklad pri kompletovaní objednávky je potrebné zobraziť informácie o objednávke, prípadne zobraziť ďalšie nadväzné informácií. Následne môžeme pridať alebo odobrať nejaké položky a následne objednávku zadať alebo ju môžeme celú zrušiť. Tieto operácie tvoria funkčné požiadavky informačného systému. Štandardný jazyk UML na definíciu funkčných požiadaviek využíva diagram prípadov užitia (Use Case diagram). Ten sa ale sústredí na funkcionality systému dostupnú užívateľského rozhrania. Preto je nutné spoliehať sa nato, že funkcionality vyplynie kombináciou diagramu prípadov užitia a diagramu tried, kde budú dostatočne dobré popísané operácie tried a ich vzájomné súvislosti. Pri procesnom modelovaní je ale tento prístup nedostatočný a môže vytvárať väčšiu prieťaž medzi business procesom a informačným systémom, a teda aj medzi business analytikmi a programátormi.

Vhodným nástrojom na popis funkčných požiadaviek pri procesnom modelovaní je **diagram dátových tokov**, známy pod skratkou **DFD** z anglického **data flow diagram**. Diagram vyjadruje statický popis požadovaných funkcií informačného systému s väzbou na tok dát od ich vzniku až po ich konzumáciu. Skladá sa zo štyroch elementov:

- **Funkcia** - predstavuje operáciu, ktorá spracováva dátu
- **Dátový tok** - predstavuje abstrakciu akejkoľvek formy presúvaných dát v systéme
- **Dátový sklad** - predstavuje abstrakciu akejkoľvek formy uloženia dát, keď v systéme dochádza k nejakej asynchronnej výmene dát
- **Terminátor** - predstavuje nejaký externý objekt systému (človek, prípadne iný systém), ktorý vytvára alebo konzumuje dátu

Pri vytváraní DFD vychádzame z operácií v diagrame tried a následne analyzujeme jednotlivé úlohy procesu (Řepa 2021) (Řepa a Svatoš 2021). Môžeme teda identifikovať aj operácie, ktoré bude treba doplniť do diagramu tried. Ukážku diagramu dátových tokov vidíme na obrázku č.2.8.



Obr. 2.8: Diagram dátových tokov (Řepa 2021, str. 99)

## 2.4 Návrh

Tak ako už bolo spomenuté, metodika MMABP sa primárne zaobera analytickými fázami. Každopádne jej princíp troch architektúr nás posúva do **technologickej vrstvy** procesnej architektúry. Pri návrhu systému dochádza k transformácií niektorých modelov, ktoré boli vytvorené v analytickej fáze.

### 2.4.1 Transformácia konceptuálnych modelov na technologické modely

Pre konceptuálne modely vytvorené pomocou metodiky MMABP existuje metodický postup transformácie z konceptuálneho do technologického modelu, ktorý navrhol Pavelčák (2021) a neskôr ho mierne revidovala Tauchmanová(2023). Pozrime sa na zjednodušený popis krokov tohto zrevidovaného metodického postupu.

1. Špecifikácia použitej technológie so zameraním na možnú zmenu granularity vykonávaných úloh
  - 1.1. analýza spôsobov vykonania procesu s prihliadnutím na využívanie rôznych technológií
  - 1.2. špecifikácia konkrétneho workflow v závislosti na konkrétnom technologickom spôsobe
  - 1.3. vytvorenie procesného diagramu pre daný proces so zohľadnením použitej technológie
2. Špecifikácia technických rolí
  - 2.1. vyhľadať aktérov či technické role v modeli business systému a jeho dokumentácií
  - 2.2. vytvoriť zoznam technických rolí/aktérov
  - 2.3. vytvoriť plavebné dráhy pre konkrétné role v procesných diagramoch
3. Špecifikácia externých udalostí
  - 3.1. identifikovať všetky udalosti v detailných procesných mapách
  - 3.2. každú udalosť zanalyzovať, či sa skutočne jedná o udalosť prichádzajúca z vonkajšieho prostredie IS a ako bude táto udalosť vykonaná (doplniť možnosť úlohy v systéme/ad hoc procesu)
  - 3.3. vytvoriť zoznam udalostí s požadovanými informáciami
  - 3.4. pridať vhodným spôsobom externé udalosti do procesných diagramov do príslušných "plavebných dráh"
4. Transformácia úloh a pridanie potrebných úloh v závislosti na alternatívnych prechodoch
  - 4.1. preskúmať všetky detailné procesné mapy a identifikovať typ všetkých úlohám,
  - 4.2. doplniť úlohy s korešpondujúcimi typmi do príslušných "plavebných dráh",
  - 4.3. doplniť novo identifikované úlohy s korešpondujúcim typom do príslušných "plavebných dráh",
  - 4.4. doplniť udalosti pre štart/y a konce procesu,
  - 4.5. doplniť spojenie vrátane príslušných brán medzi jednotlivými úlohami a

- udalostami,
- 4.6. doplniť podmienky/popisy u XOR brán.
  5. Určenie vstupov a výstupov pre každú úlohu procesu
    - 5.1. pre každý procesný diagram pripraviť tabuľku
    - 5.2. do tabuľky vyplniť názov úlohy a identifikovať a zapísat potrebné vstupy a výstupy
    - 5.3. prípadne doplniť zodpovedné roly danej úlohy
    - 5.4. nejasnosti si vyjasniť s vlastníkom procesu
  6. Aktualizácia diagramu tried
    - 6.1. prejsť všetky procesné diagramy a skontrolovať, prípadne doplniť, chýbajúce triedy alebo atribúty vrátane dátových typov,
    - 6.2. do diagramu tried doplniť stavové premenné,
    - 6.3. skontrolovať, prípadne pridať, technické role/aktéri v diagrame tried
    - 6.4. skontrolovať, prípadne doplniť, dátové typy, povinnosti dátových typov a väzieb medzi jednotlivými triedami
  7. Špecifikácia notifikácií
    - 7.1. identifikovať úlohy, v ktorých sú notifikácie poslané,
    - 7.2. diskutovať znenie textov notifikácií, vrátane odosielateľov a príjemcov, s vlastníkom procesov, prípadne s klúčovými užívateľmi,
    - 7.3. vytvoriť tabuľku pre každú notifikáciu, ktorá obsahuje potrebné informácie na základe charakteru notifikácie.
  8. Pridanie ďalšieho popisu či poznámok

Z týchto krokov vyplýva, že vznikne:

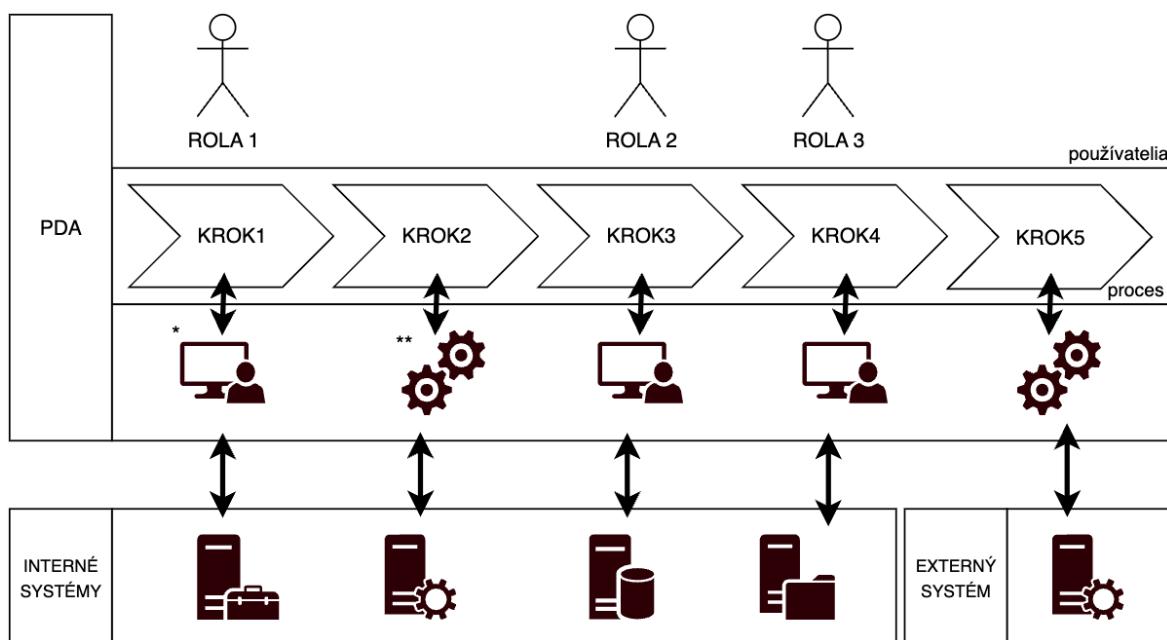
- technologický procesný diagram (podla štandardu BPMN 2.0)
- technologický diagram tried
- zoznam technických rolí
- zoznam úloh so vstupmi a výstupmi
- zoznam udalostí
- tabuľky notifikácií

## 2.5 Implementácia

Originálny Royce-ov model hovoril o fáze kódenie, procesne riadená architektúra je ale výborným príkladom toho, prečo sa fáza premenovala na implementácia. Písanie kódu totiž nie je jediná činnosť pri vytváraní informačného systému pre procesne riadený podnik. Nachádzame sa v poslednej **implementačnej vrstve** trojvrstevnej procesnej architektúry metodiky MMABP. Skôr ako sa presunieme ku popisu vykonávaných krokov si zavedieme niekoľko ďalších dôležitých pojmov.

### 2.5.1 Procesne riadená aplikácia

Definícia procesne riadenej aplikácie (PDA) hovorí o tom, že je to aplikácia podporuje komplexné podnikové procesy. Zahŕňa funkčné, systémové a organizačné obmedzenia a používa dát a funkcie z rôznych iných aplikácií a systémov (Stiehl 2014). Zjednodušene by sme mohli povedať, že PDA prepája funkcionalitu viaceru podnikových systémov a môže byť formou implementácie procesne riadenej architektúry. Systematické prepojenie podnikových systémov umožňuje vykonávať manuálne činnosti zamestnancom napriek rôznymi oddeleniami či pracovnými rolami v postupnosti a súlade s podnikovými procesmi. V súčasnosti sa veľké množstvo činností podnikových procesov vykonáva automaticky. PDA teda okrem činnosti vyžadujúcich aktivitu zamestnanca zapája do podnikového procesu aj automatizované činnosti. Pre lepšie pochopenie môžeme vidieť na obrázku 2.9 náčrt priebehu procesu v PDA.



Obr. 2.9: Náčrt priebehu procesu v PDA

## **2.5.2 Workflow a workflow management systém**

Automatizácia podnikového procesu, ktorú umožňujú procesne riadené aplikácie sa označuje pojmom **workflow**. Procesne riadená aplikácia je všeobecnejší pojem popisujúci koncept prepájania činnosti v podniku v súlade s podnikovým procesom. Workflow definuje (2019) ako "automatizácia podnikového procesu obsahujúcu aktivity usporiadane v ich obvyklom poradí, ktoré sú vykonávané v jednom systéme". Dalo by sa povedať, že je technickejší popis toho, akým spôsobom sa implementuje procesne riadená aplikácia.

Workflow implementuje v tzv. **workflow management systéme (WfMS)**. Je to systém, ktorý umožňuje definovať, vytvárať a spravovať vykonávanie workflow vo forme aplikácie (Weske 2019). Inými slovami, je to systém umožňujúci vytvárať a spravovať automatizovaný podnikový proces.

## **2.5.3 Implementácia workflow**

Kedže sa workflow implementuje v WfMS systéme, znamená to, že implementačná vrstva už nie je nezávislou vrstvou. Pri jej tvorbe vychádzame z implementačne nezávislých vrstiev (konceptuálne a technologické modely) a vytvárame implementačne závislé modely, ktoré sú použiteľné iba v konkrétnom WfMS.

V sekciu 6.2 navrhujeme metodický postup transformácie technologického modelu do implementačného modelu pre workflow management systém Camunda 8.



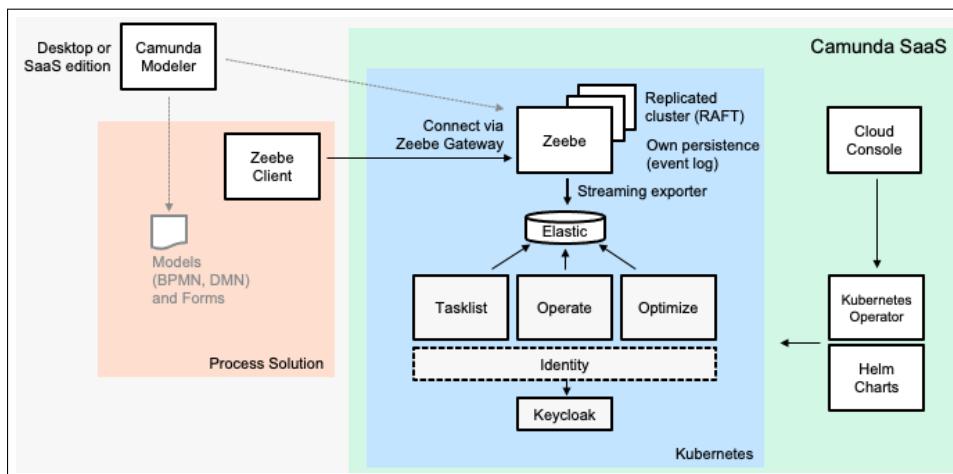
# 3. Camunda

Camunda Platform 8 je workflow management systém, slúžiaci na orchestráciu komplexných business procesov ([Camunda Platform 8 Docs 2023](#)). Je navrhnutý tak, aby bol dobre horizontálne škálovateľný, mal vysokú dostupnosť a odolnosť voči chybám ([Concepts 2023](#)).

Verzia 8 bola vydaná v apríli roku 2022. Nahradila tak verziu 7, ktorá vznikla už v roku 2013. Podpora staršej verzie je deklarovaná ešte na niekoľko rokov, ale presný rok nie je stanovený. Pri verzii 8 došlo k niekoľkým vylepšeniam, no asi najdôležitejšou bola zmena architektúry, ktorej umožňuje dosahovať deklarované vlastnosti platformy (DEEHAN 2023).

## 3.1 Komponenty a užívateľské zohrania

Architektúra tejto platformy (obrázok č.3.1) sa skladá z niekoľkých komponentov. Pre prácu s platformou ale nepotrebuje poznáť technické detaile architektúry. Predstavíme si teda iba tie najdôležitejšie komponenty, ktoré majú často podobu užívateľského rozhrania s ktorým pracujeme. Nie všetky komponenty sú ale dostupné vo všetkých inštaláčnych respektíve cenových variantoch. O tom, ktoré sú alebo nie sú dostupné sa dočítame v kapitole 3.2.



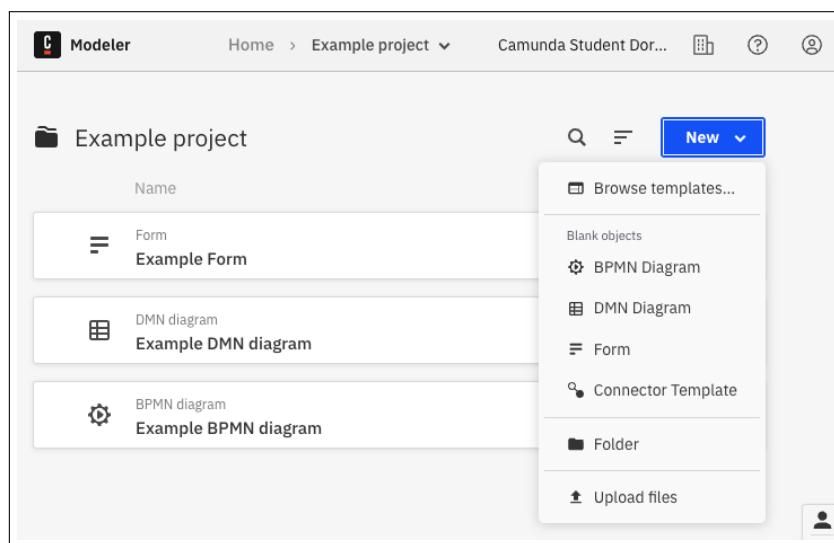
Obr. 3.1: Architektúra platformy Camunda 8 ([Overview Components 2023](#))

### 3.1.1 Camunda Modeler

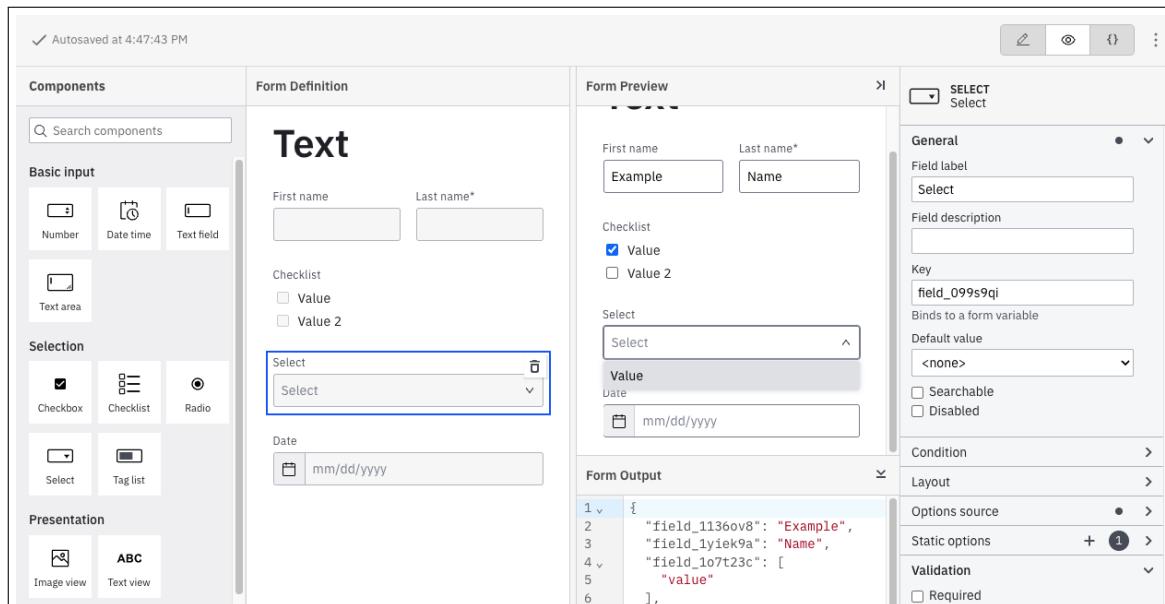
Túto komponentu používame pri modelovaní. Skladá sa z troch rôznych užívateľských rozhraní, ktoré sa používajú na vytváranie BPMN diagramov procesov, DMN diagramov a formulárov. Práca s nimi je veľmi intuitívna. Samotné jadro rozhraní sa vyvinulo do samostatného open source projektu s názvom BPMN.iO, a používa sa v rôznych iných workflow management nástrojoch.

## Web Modeler - webová verzia

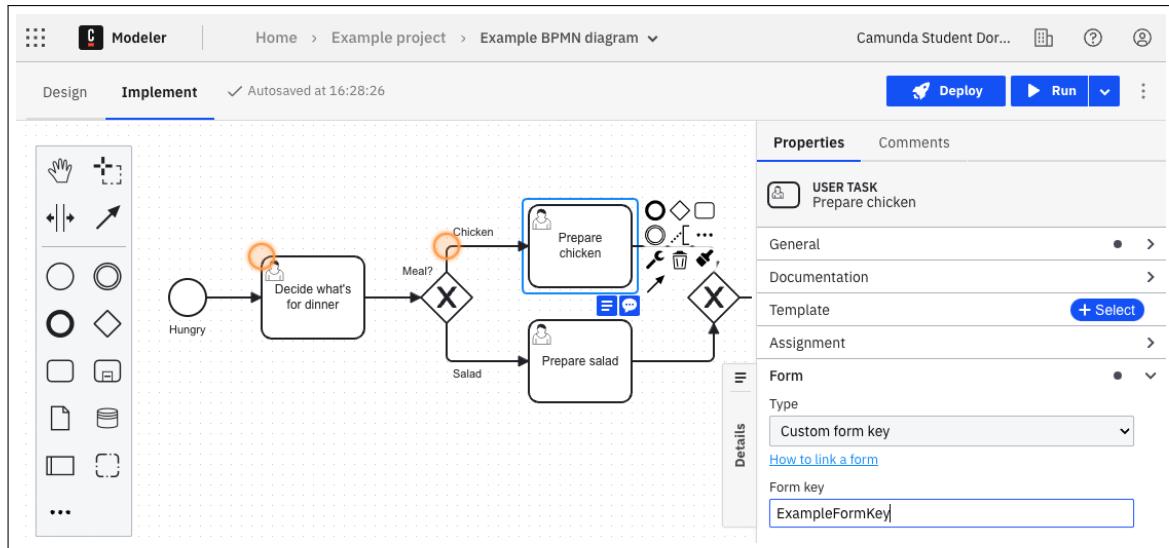
Táto verzia je dostupná hlavne pre platené verzie platformy. Umožňuje vytvárať prehľadne štruktúrovaný projekt (obrázok č.3.2). Po vytvorení súboru vybraného typu (napr. formulár) a následnom otvorení sa dostaneme priamo do požadovaného rozhrania na úpravu. Rozhranie pre tvorbu formulárov môžeme vidieť na obrázku č.3.3. Výhodou webovej verzie je jednoduché prepojenie medzi rôznymi typmi súborov. Na obrázku č.3.4 vidíme rozhranie pre tvorbu BPMN diagramov. Pri napojení formuláru na *user task* stačí zadať klíč vytvoreného formulára. Z tohto prostredia je možné diagram nasadiť prípadne aj spustiť inštanciu daného procesu.



Obr. 3.2: Štruktúra projektu v Camunda Web Modeler



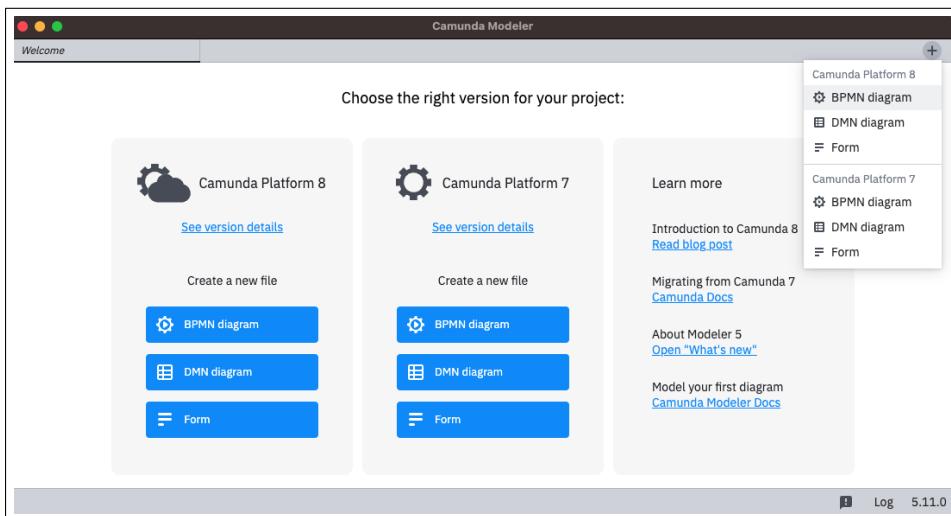
Obr. 3.3: Tvorba formuláru v Camunda Web Modeler



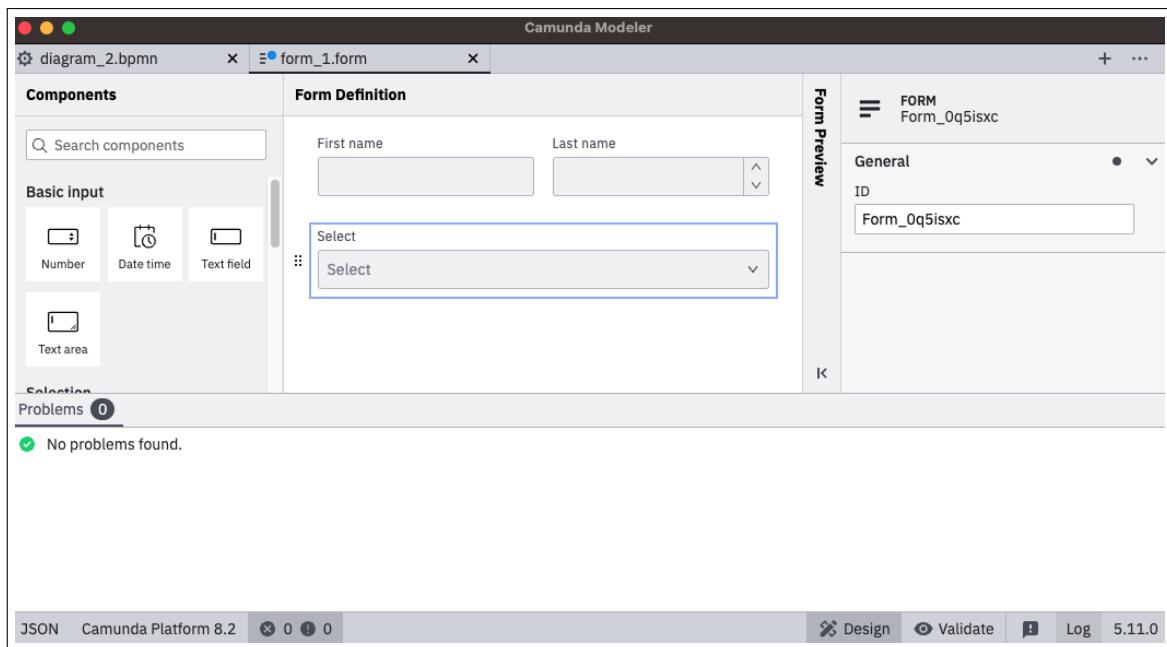
Obr. 3.4: Tvorba BPMN diagramu v Camunda Web Modeler

### Desktop Modeler - desktopová verzia

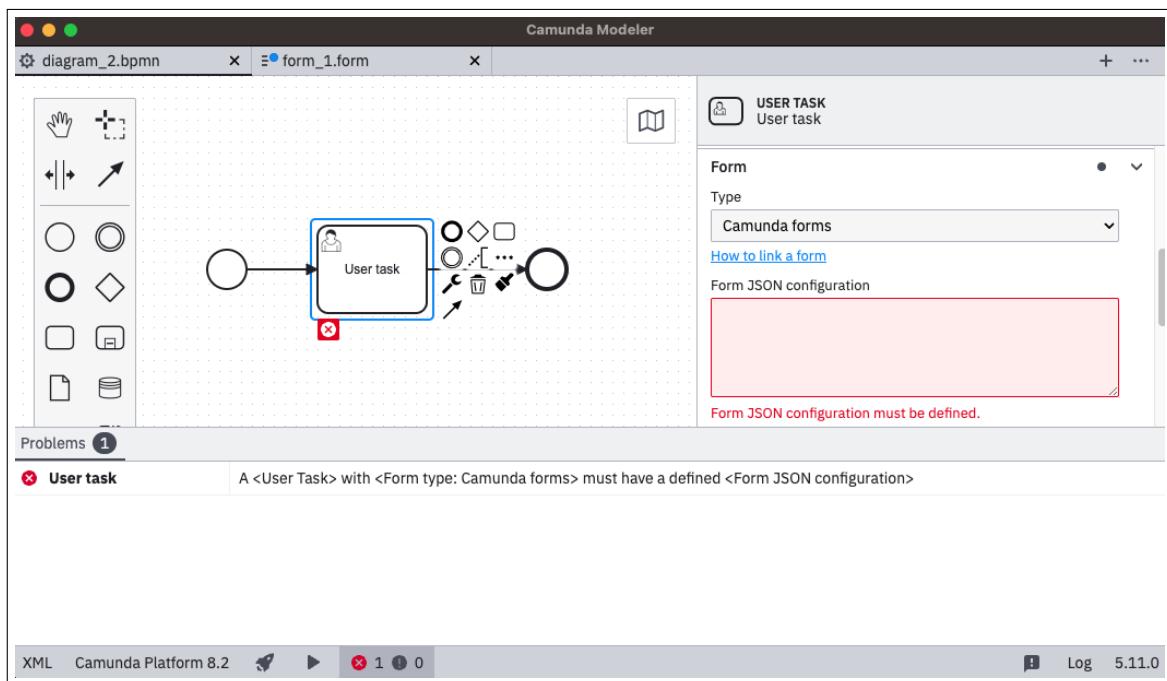
Táto verzia sa od webovej výrazne nelíši pri tvorbe diagramov a formulárov. Líši sa však pri spravovaní týchto súborov a prepájaní diagramov a formulárov. Aplikácia priamo neposkytuje správu súborov, ponúka len možnosť vytvorenie vybraného typu súboru (obrázok č. 3.5) ale miesto uloženia, a teda aj celá správa projektových súborov je na nás. Tvorba BPMN diagramu je takmer rovnaká ako vo webovej verzii (obrázok č.3.7). Ale pri pokuse spojiť formulár s BPMN diagramom nám bohužiaľ nebude správne fungovať prepojenie cez formulárový klíč. Jediná možnosťou je po vytvorení formuláru skopírovať JSON konfiguráciu, ktorú nájdeme pod tlačidlom, ktoré je umiestnené na spodnej liště modeléru (obrázok č.3.6) a prilepíme ju priamo do *Form JSON configuration* pola *user task* elementu, ktoré vidíme na obrázku č.3.7.



Obr. 3.5: Štruktúra projektu v Camunda Desktop Modeler



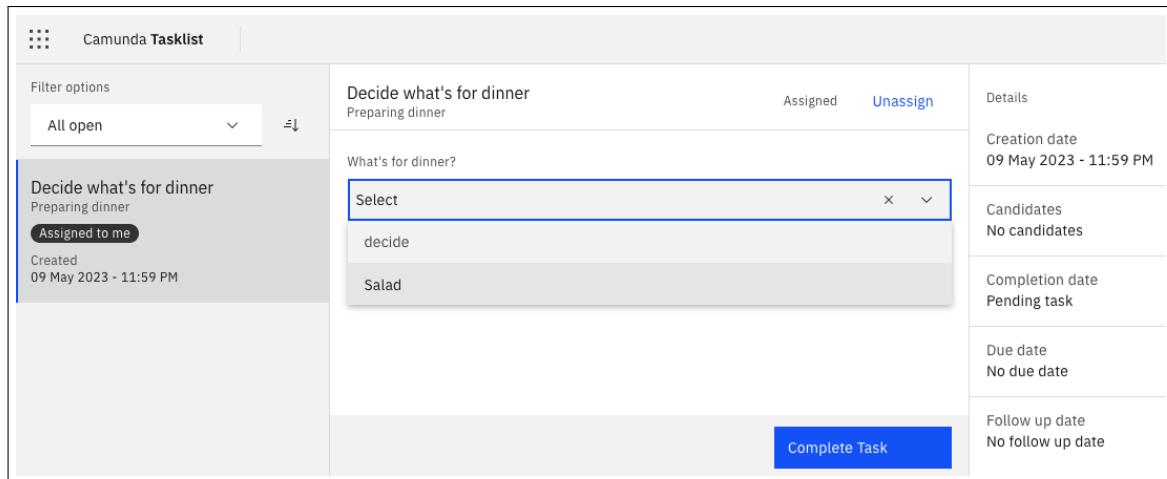
Obr. 3.6: Tvorba formuláru v Camunda Desktop Modeler



Obr. 3.7: Tvorba BPMN diagramu v Camunda Desktop Modeler

### 3.1.2 Tasklist

Toto rozhranie slúži na prácu s činnosťami, ktoré sú v procese v podobe *user task* určené človeku (obrázok č.3.8). Užívateľovi sa priradením úlohy otvorí formulár, ktorý môže vyplniť a následne ukončiť túto úlohu.



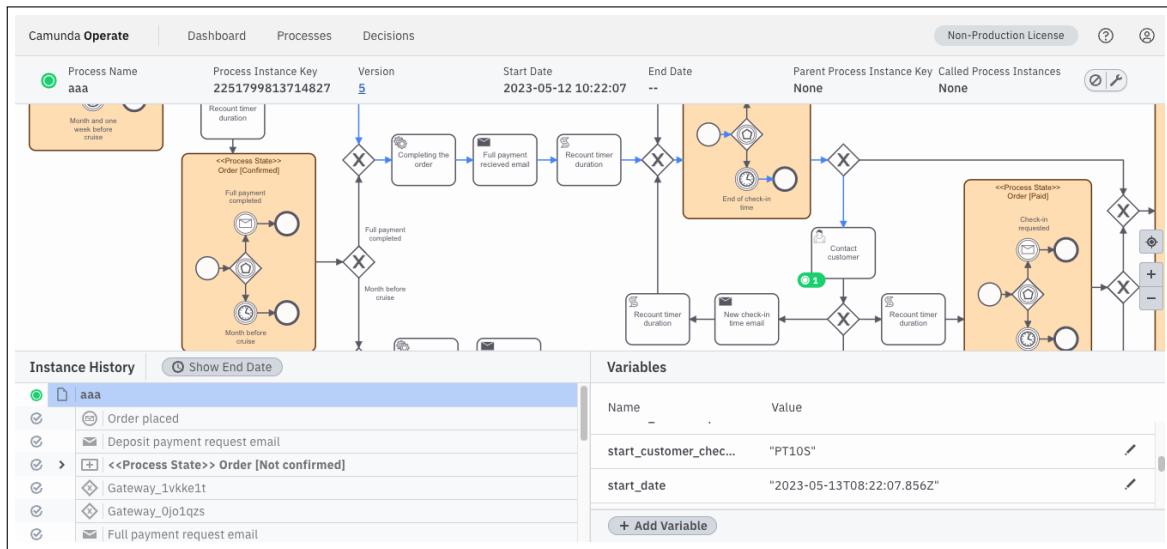
Obr. 3.8: Camunda Tasklist

### 3.1.3 Operate

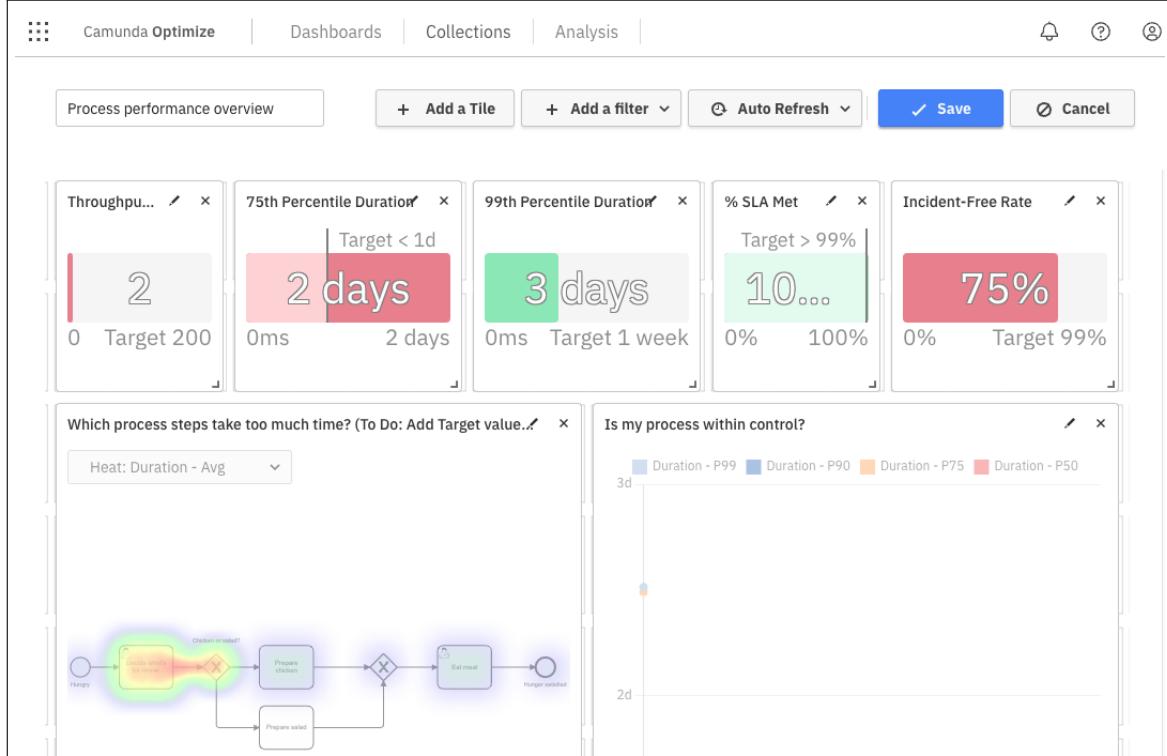
Toto rozhranie slúži na monitorovanie a riešenie problémov s inštanciami procesu. Obsahuje zoznam bežiacich ale aj ukončených inštancií. Umožňuje zobraziť v akom stave sa konkrétna inštancia nachádza a aké sú hodnoty premenných, ktoré sa v nej používajú (obrázok č.3.9). Je možné meniť hodnoty premenných, či zrušiť inštanciu procesu. Je to dobrým pomocníkom pri vytváraní a testovaní procesu ale aj pri hľadaní chýb v produkčnom prostredí.

### 3.1.4 Optimize

Názov komponenty nám napovedá, že jej cieľom je optimalizácia. Ide o business intelligence nástroj, ktorý umožňuje zbierať informácie z inštancií procesov, ktoré môžu slúžiť na reportovanie a následnú optimalizáciu procesov.



Obr. 3.9: Camunda Operate



Obr. 3.10: Camunda Optimize

### 3.1.5 Zeebe engine a Zeebe klient

Hlavným motorom, ktorý dokáže namodelovaný workflow v podobe BPMN diagramu spustiť je workflow engine s názvom Zeebe. Je napísaný v programovacom jazyku Java.

V teoretickej časti sme sa bavili o automatizovaných činnostiach v procese, prepájaní viacerých systémoch a servisne orientovanej architektúre. Na toto prepájanie používame *service task* element v BPMN diagrame. Komunikácia medzi Zeebe enginom a nejakým iných systémom môže prebiehať prostredníctvom tzv. Zeebe klientov. Dokumentácia ([Working with APIs and tools 2023](#)) uvádza troch oficiálnych klientov, pre tieto technológie:

- CLI konzola
- programovací jazyk Go
- programovací jazyk Java

Dokumentácia ale odkazuje aj na komunitné nástroje pre rôzne iné technológie ako programovací jazyk Python, Ruby, Rust či platformu NodeJS (programovací jazyk JavaScript), ktorá bola použitá aj v praktickej časti tejto práce. Pomocou klientov teda môžeme implementovať servisnú úlohu, alebo vyvolať nejakú udalosť či nasadiť alebo spustiť proces. Ukážku použitia NodeJS klienta môžeme vidieť vo výpise kódu 3.1.

```
1 const { ZBClient } = require('zeebe-node')
2 const zbc = new ZBClient('localhost', { loglevel: 'DEBUG' });
3
4 zbc.createProcessInstance('process-name');           //Spusti instanciu procesu
5
6 zbc.createWorker({          //Vytvorí 'job work' na spočítanie dvoch čísel
7   taskType: 'count-two-number',
8   taskHandler: (job) => {
9     const a = job.variables.a; //vstupné premenne z procesu 'job.variables'
10    const b = job.variables.b;
11    job.complete({          //ukončí pracu s novou výstupnou premennou 'result'
12      result: a + b
13    });
14  },
15});
```

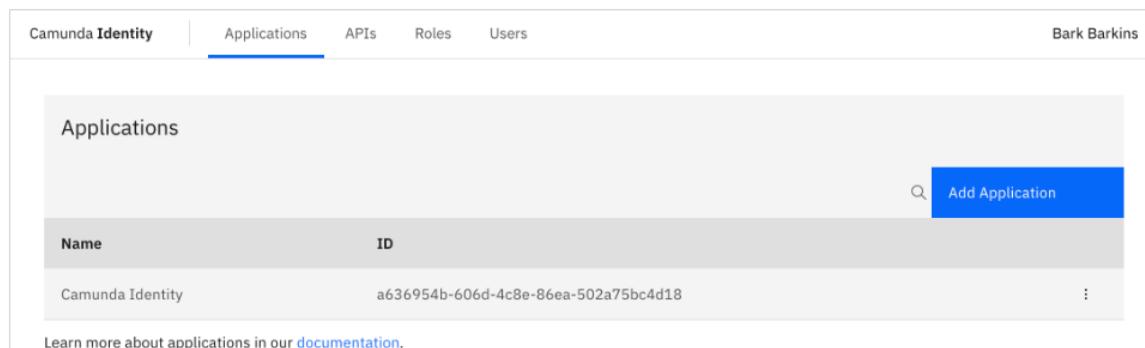
  

```
16
17 const orderId = 15;
18 zbc.publishMessage({           //Spusti udalosti typu 'message'
19   name: "PAYMENT_COMPLETED",
20   correlationKey: orderId,
21 })
```

Výpis 3.1: Zeebe klient pre NodeJS

### 3.1.6 Identity

Poslednou pre nás zaujímavou komponentou je Identity, ktorá slúži na autentizáciu a autorizáciu. Nastavujú sa v nej prístupové klíče pre API, užívatelia a ich role. Bohužiaľ táto komponenta je dostupná len pre platené typy inštalácií. Preto sa s ňou v praktickej časti práce nestretneme.



The screenshot shows the Camunda Identity web interface. At the top, there is a navigation bar with tabs: 'Camunda Identity' (selected), 'Applications' (highlighted in blue), 'APIs', 'Roles', and 'Users'. To the right of the tabs, the name 'Bark Barkins' is displayed. Below the navigation bar, the title 'Applications' is centered. On the right side of the main area, there is a blue button labeled 'Add Application' with a magnifying glass icon. The main content area displays a table with two columns: 'Name' and 'ID'. A single row is present, showing 'Camunda Identity' in the Name column and 'a636954b-606d-4c8e-86ea-502a75bc4d18' in the ID column. At the bottom left of the table, there is a link: 'Learn more about applications in our [documentation](#)'.

Obr. 3.11: Camunda Identity ([Identity 2023](#))

## 3.2 Typy inštalácie a cena

Camunda poskytuje dva typy inštalácie platformy. Prvým typom je *SaaS - software as a service* a druhým typom je *Self-Managed*. Povieme si čo tieto typy inštalácií ponúkajú a aké sú ceny (v čase písanie práce) za ich využívanie ([Camunda Platform 8 Pricing 2023](#)).

### 3.2.1 SaaS Camunda

Pri výbere SaaS typu nie je nutná žiadna inštalácie vývojového alebo produkčného prostredia. K dispozícii je webové aplikácia, ktorá v sebe zahŕňa všetky typy užívateľských rozhraní. Existujú tri cenové varianty. Prvá varianta, ktorá je zadarmo ale poskytuje iba možnosť vytvárať BPMN a DMN modely. Neumožňuje spúštať inštancie procesu, preto nemá veľké praktické využitie. Pozrime sa na ďalšie cenové varianty.

#### Professional

Cena tejto varianty začína na 49 dolároch mesačne. Ponúka možnosť pre prácu 10 užívateľov, spustenie 50 inštancií procesu a 50 rozhodovacích inštancií mesačne. Toto množstvo sa dá za príplatok navýšiť. Technická podpora je k dispozícii 5 dní v týždni.

#### Enterprise

Cena tejto varianty je iba na vyžiadanie. Ponúka neobmedzené množstvo užívateľov, inštancií procesu a rozhodovacích inštancií. Technická podpora je k dispozícii 24 hodín, 7 dní v týždni.

### **3.2.2 Self-managed Camunda**

Pri výbere Self-managed typu je nutná vlastná inštalácia. Existujú dva inštalačné varianty:

- Pre lokálny vývoj pomocou Docker Compose
- Pre produkčné nasadenie pomocou Kubernetes

Existujú dva cenové varianty Self-managed typu.

#### **Free**

Táto variantu je možné používať zdarma. Na produkčné použitie je ale možné využívať iba samotný Zeebe engine a desktopovú verziu modelera. Prostredia Operate, Tasklist a Optimize sú dostupné len na neprodukčné použitie.

#### **Enterprise**

Táto varianta má rovnako ako SaaS Enterprise varianta cenu len na vyžiadanie. Ponúka aj takmer zhodný balík služieb.



# 4. Transformácia konceptuálneho modelu na technologický model

V tejto kapitole budeme overovať postup transformácie konceptuálneho modelu na technologický model, ktorý vytvoril Pavelčák a revidovala Tauchmanová. Pred touto transformáciou si ukážeme konceptuálny model fiktívnej spoločnosti, ktorá poskytuje prenájom námorných rekreačných plavidiel (charter lodí) a zavádzajú procesné riadenie. Následne vykonáme transformáciu do technologického modelu. Potom zhodnotíme daný metodický postup a navrhнемe prípadne úpravy alebo doplnenie postupu.

## 4.1 Konceptuálny model

### Globálny model systému procesov

Na procesnej mape vidíme (obrázok č.4.1), že pre spoločnosť sú najdôležitejšie business funkcie charter námorného rekreačného plavidla, správa lode a správa mimoriadnych udalostí. Celý proces začína od prejavenejho záujmu zákazníka (vytvorením objednávky) až po uskutočnený nájom. Správa lode sa zaoberá úkonmi, ktoré je nutné absolvovať pred a po samotnej plavbe. Správa mimoriadnych udalostí rieši situácie, ktoré môžu vzniknúť počas plavby alebo po ukončení plavby.

### Detailný model procesu

Kľúčový proces s názvom **Prenajatie lode** (obrázok č.4.2 a č.4.3) popisuje činnosti od vytvorenia objednávky až po ukončenie objednávky. Zároveň je orchestračným prvkom pre podporné procesy, ktoré sú z neho volané.

Podporný proces s názvom **Nalodenie** (obrázok č.4.4) popisuje činnosti od príchodu zákazníka v dohodnutý deň nalodenie, cez kontrolu potrebných dokladov k plavbe, kontrolu lode zákazníkom až po riešenie drobných problémov s loďou, zistených pri zákazníckej kontrole. Tento podporný proces musí absolvovať každý zákazník. Pre spoločnosť je cieľom zistiť, či má zákazník všetky požadované dokumenty pre plavbu. Pre zákazníka je cieľom skontrolovať aktuálny stav lode.

Podporný proces s názvom **Vylodenie** (obrázok č.4.5) popisuje činnosti od návratu lode do prístavu, ktoré sú sústredené na kontrolu stavu lode po plavbe. Tento podporný proces musí absolvovať každý zákazník. Pre podnik je cieľom zistiť, či nedošlo k poškodeniu lode. Pre zákazníka je naopak cieľom preukázať, že loď odovzdáva v rovnakom stave, ako ju prevzal.

Podporný proces s názvom **Riešenie havárie** (obrázok č.4.6) môže byť spustený počas priebehu plavby a popisuje činnosti, ktorými sa rieši vzniknutá havária. Ak je to možné je potrebné pri havárií dopraviť loď a hlavne posádku lode do prístavu. Po ukončení tohto procesu sa spúšťa podporný proces *Vylodenie*, ak nedošlo k potopeniu plavidla.

Podporný proces s názvom **Riešenie škodovej udalosti** (obrázok č.4.7) sa spúšťa ak boli pri vylodení zistené škody alebo pri havárií došlo k potopeniu lode. Podľa výšky škody rieši strhnutie zákazníckej zálohy prípadne poistnú udalosť.

### **Globálny model systému objektov**

Diagram tried (obrázok č.4.8) popisuje hlavne pre podnik najdôležitejšie business objekty, ktoré sa v procesoch vyskytujú ako napríklad *Objednávky* či *Plavba*. Ale zároveň obsahuje aj menej dôležité ale stále potrebné objekty ako napríklad *Kapitán* či *Preberací protokol*.

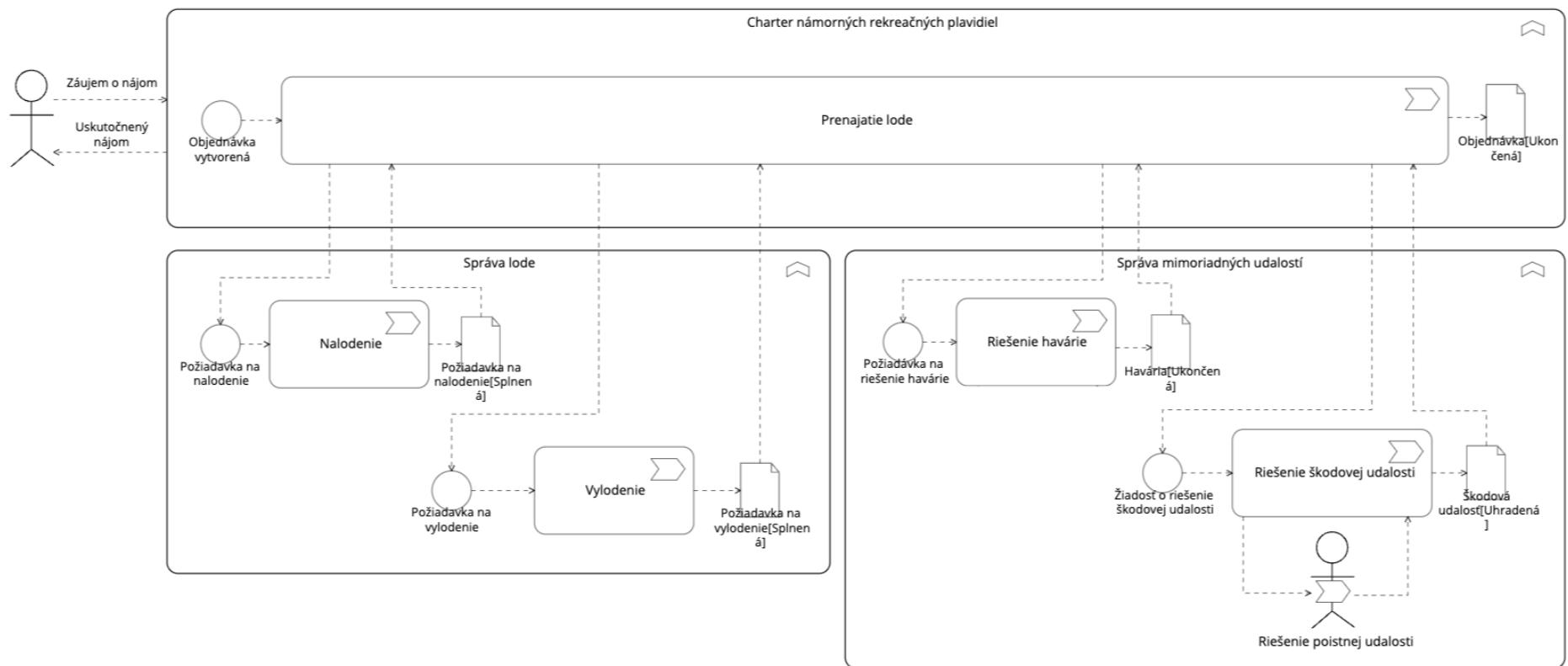
### **Detailný model objektov**

Detailný model objektov popisuje životný cyklus nasledujúcich, pre business dôležitých, objektov:

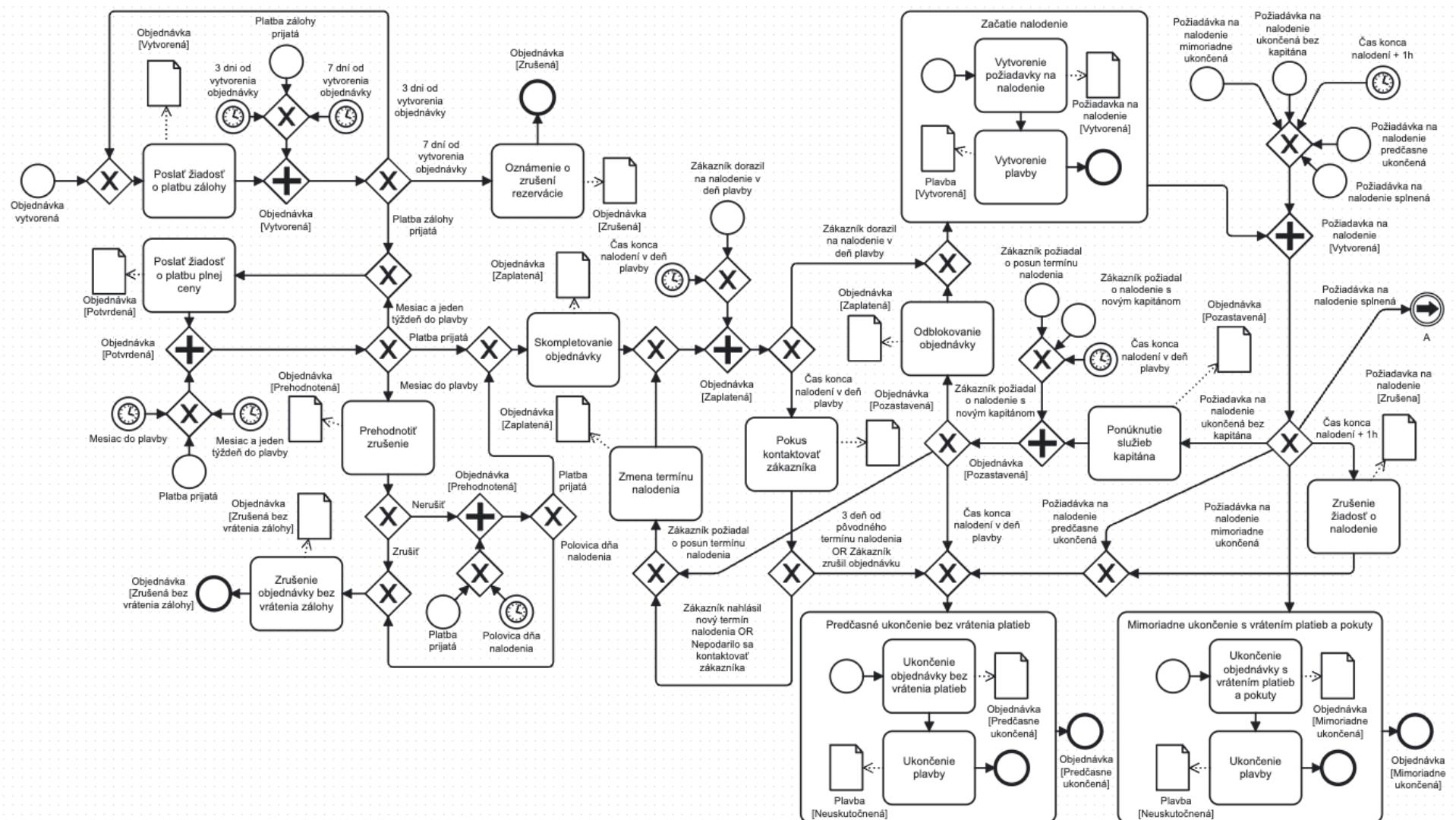
- Objednávka - obrázok č.4.9
- Plavba - obrázok č.4.10
- Požiadavka na nalodenie - obrázok č.4.11
- Požiadavka na vylodenie - obrázok č.4.12
- Preberací protokol a Odovzdávací protokol - obrázok č.4.13
- Havária - obrázok č.4.14
- Škodová udalosť - obrázok č.4.15
- Loď - obrázok č.4.16

### **Diagram dátových tokov**

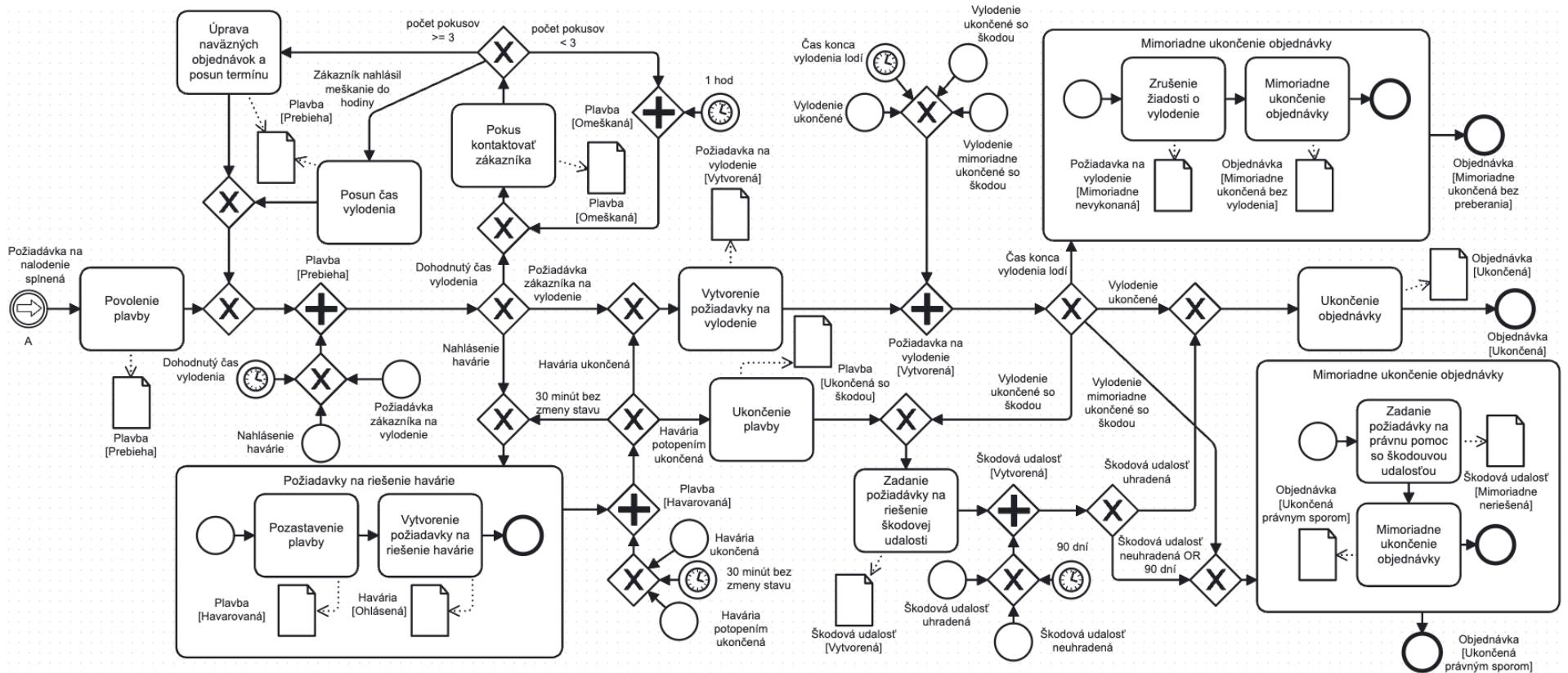
Na ukážku transformácie z technologického do implementačného modelu budeme používať klúčový proces a podporný proces Nalodenie. Vytvorili sme tak diagramy dátových tokov pre tieto dva, pre nášu prácu najdôležitejšie, procesy. Na obrázku č.4.17 vidíme DFD klúčového procesu a na obrázku č.4.18 vidíme DFD podporného procesu Nalodenie.



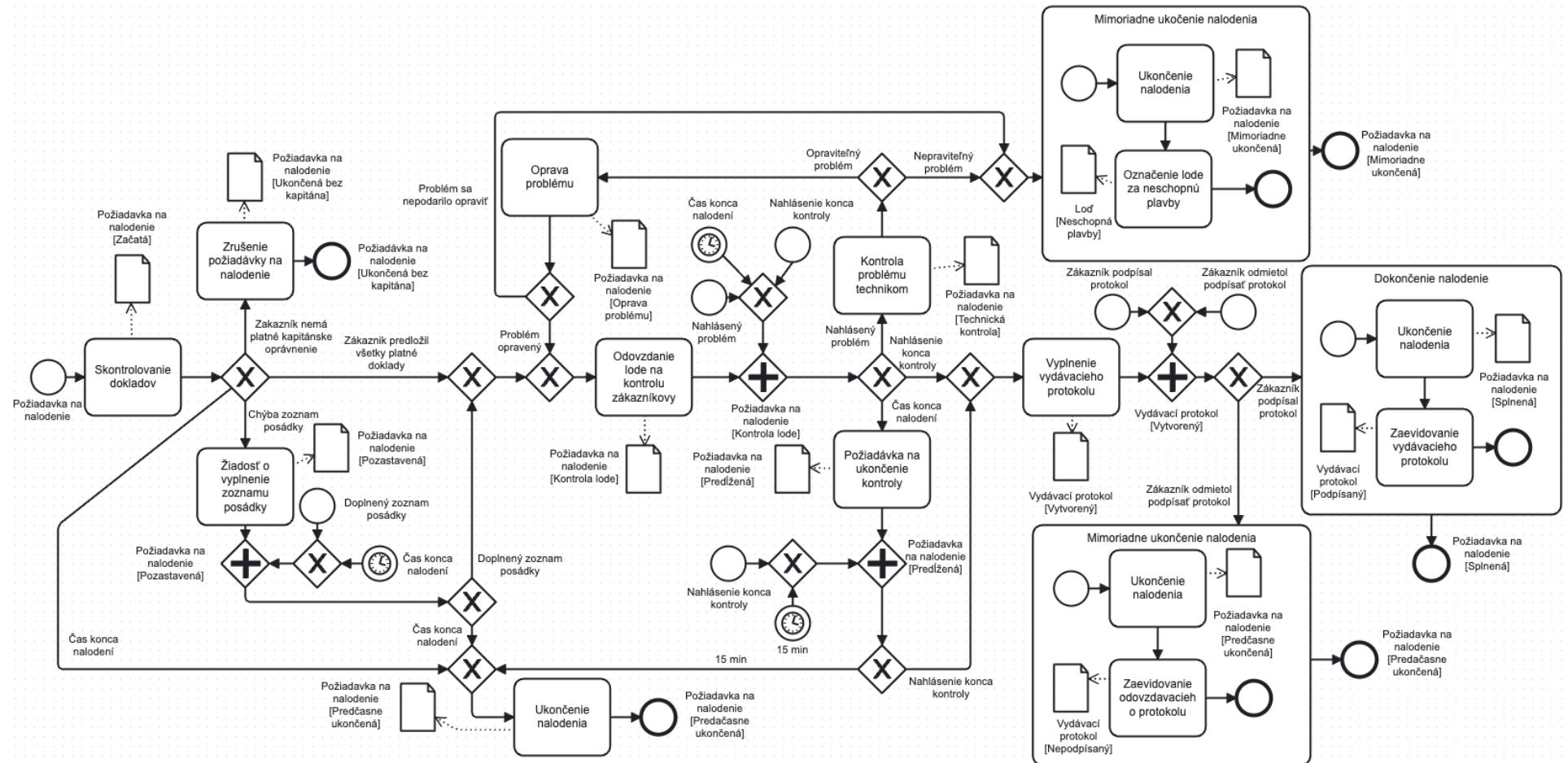
Obr. 4.1: Konceptuálny model: procesná mapa



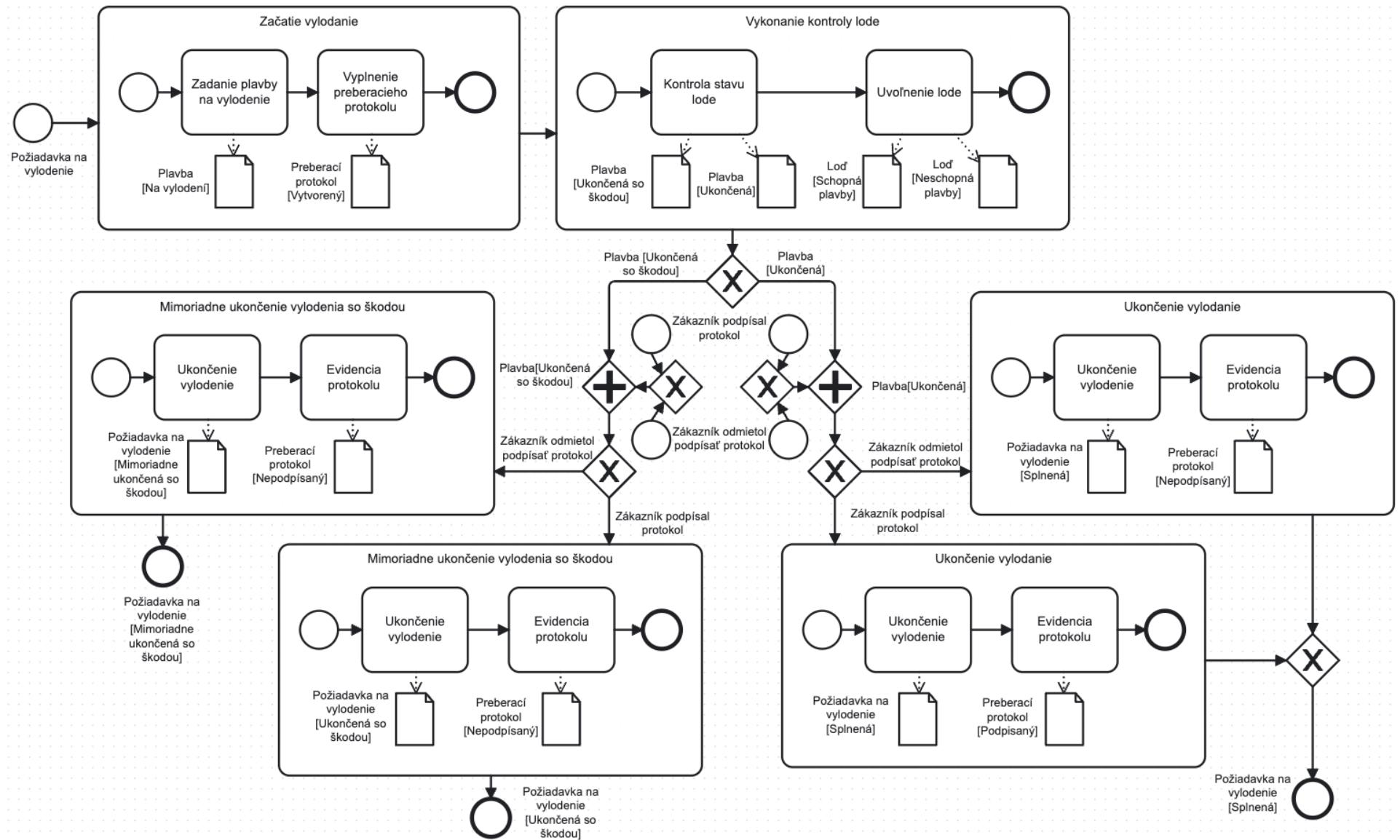
Obr. 4.2: Konceptuálny model: diagram klúčového procesu Prenajatie lode č.1



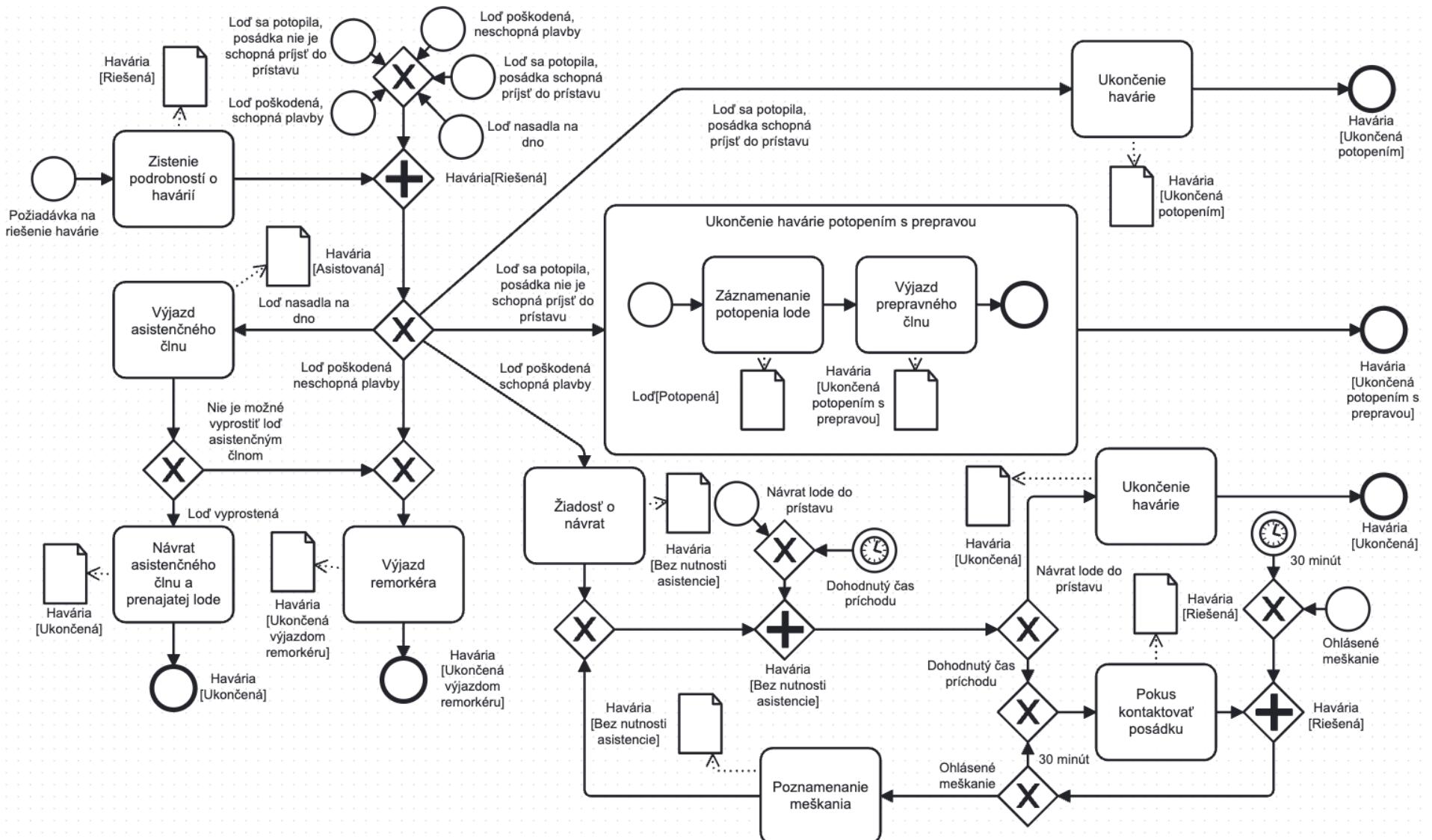
Obr. 4.3: Konceptuálny model: diagram kľúčového procesu Prenajatie lode č.2



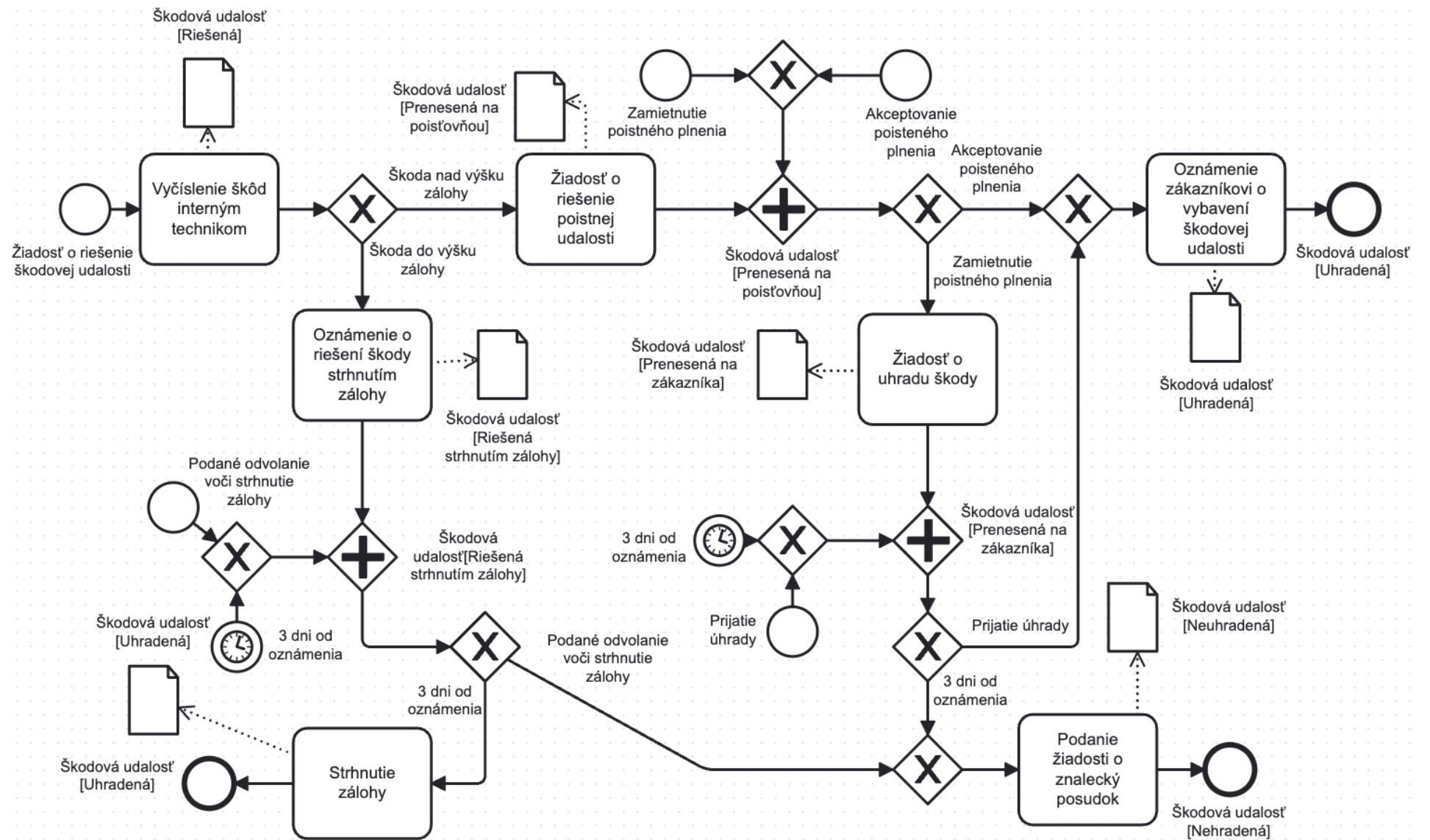
Obr. 4.4: Konceptuálny model: diagram podporného procesu Nalodenie



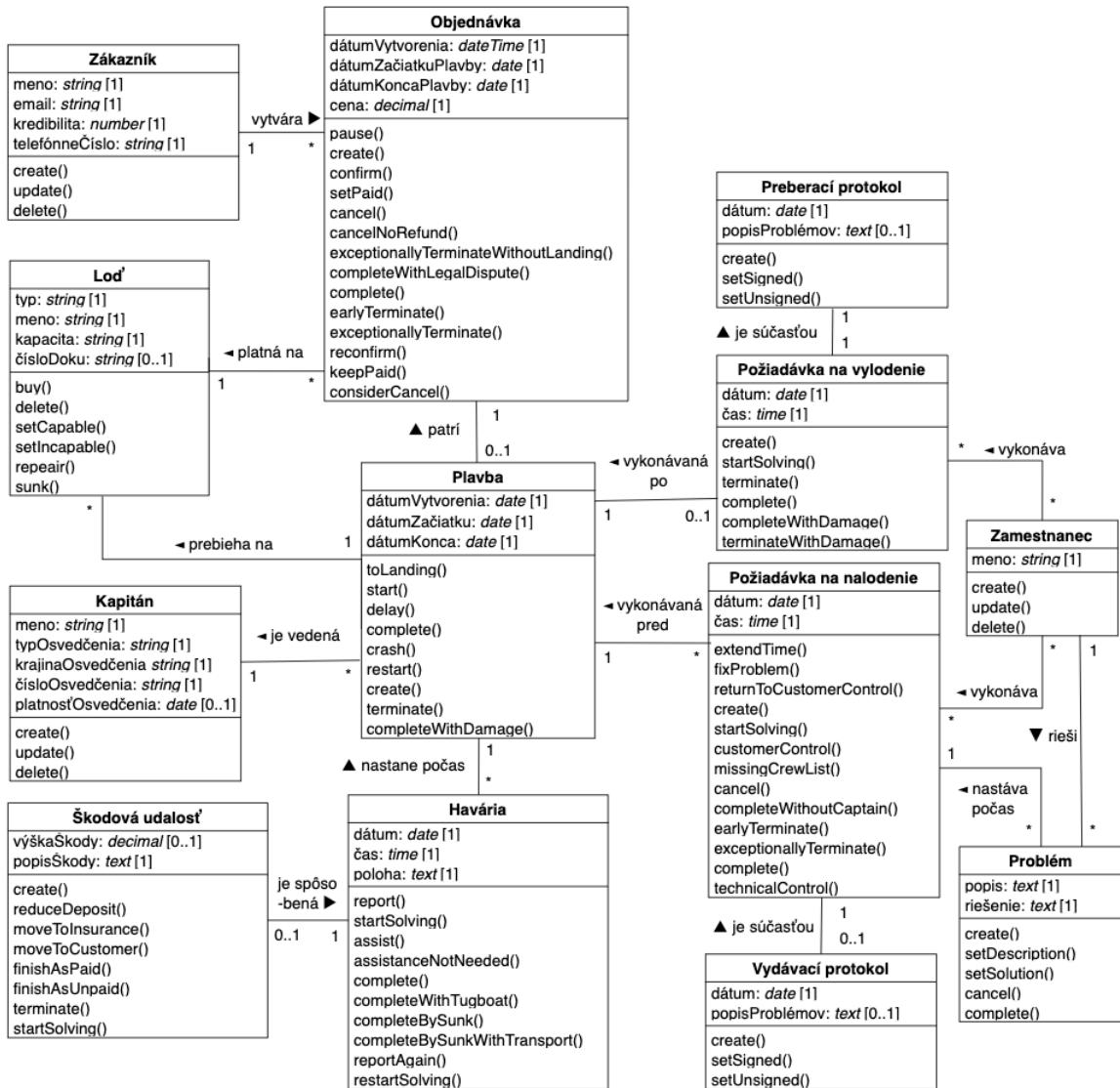
Obr. 4.5: Konceptuálny model: diagram podporného procesu Vylodenie



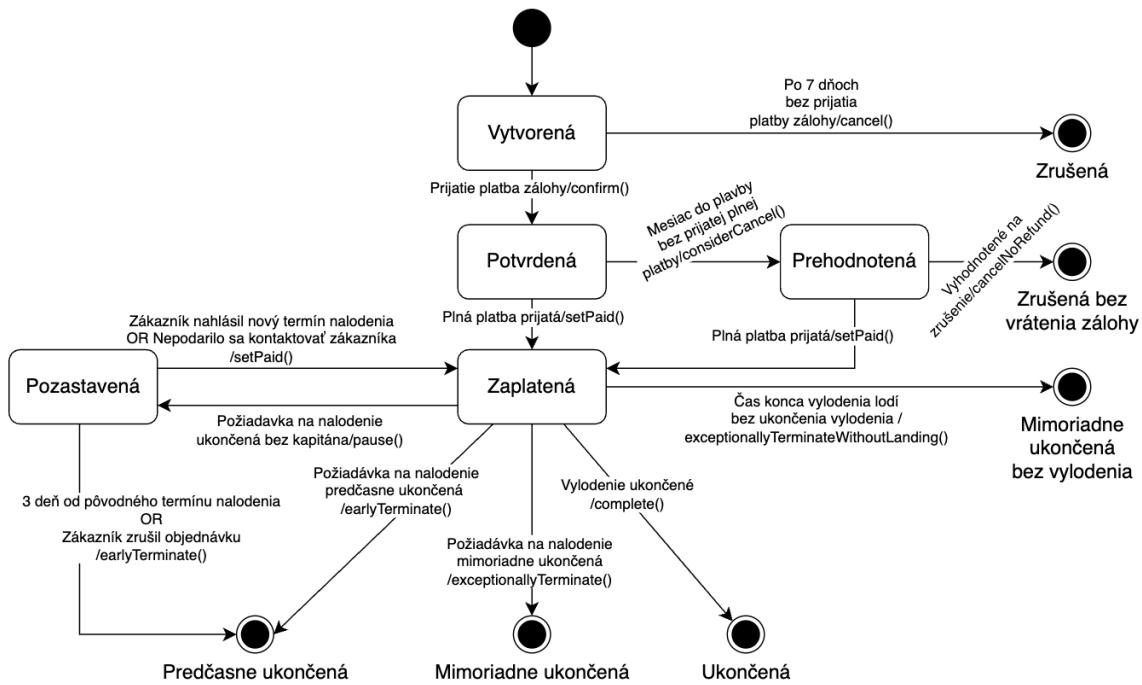
Obr. 4.6: Konceptuálny model: diagram podporného procesu Riešenie havárie



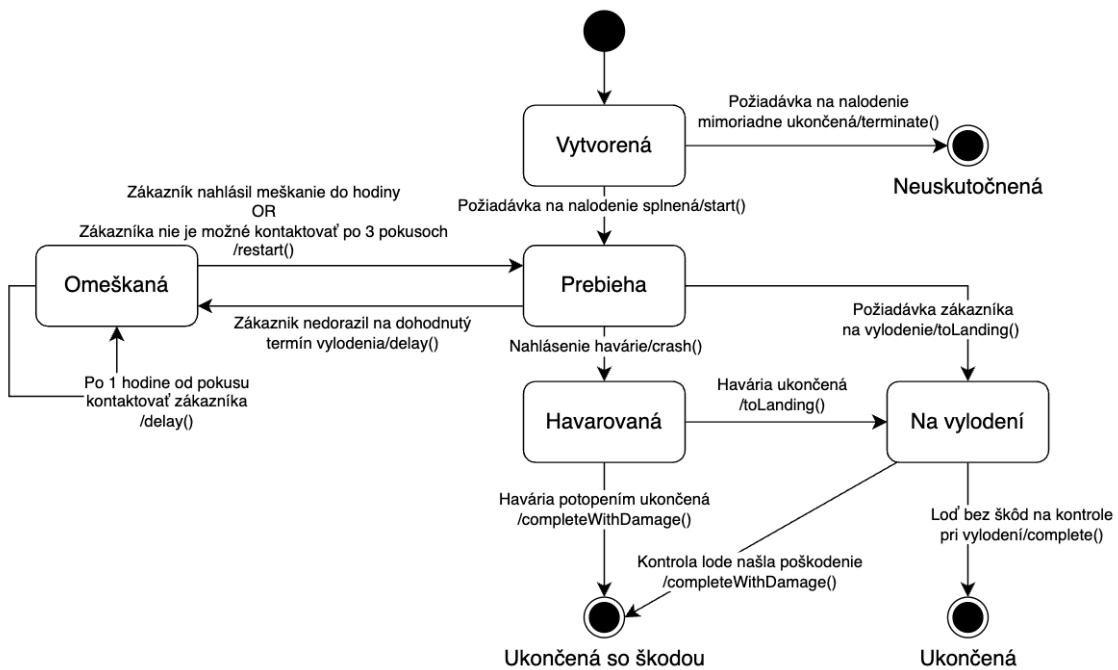
Obr. 4.7: Konceptuálny model: diagram podporného procesu Riešenie škodovej udalosti



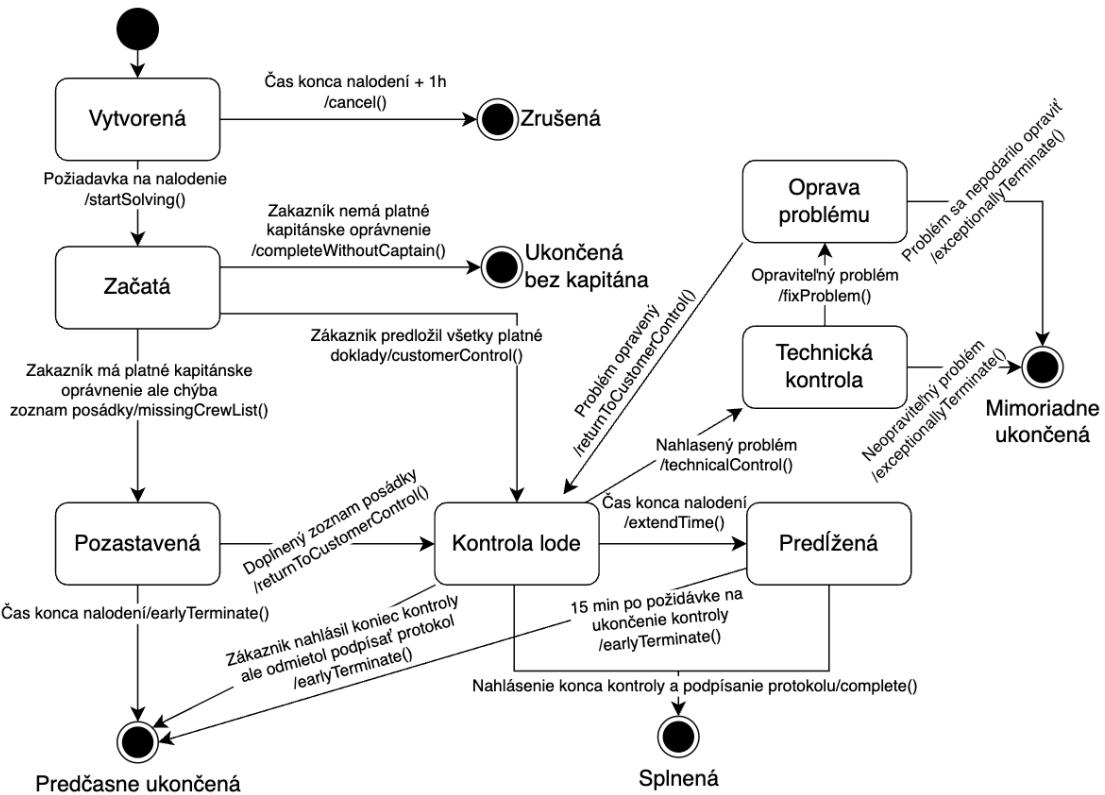
Obr. 4.8: Konceptuálny model: diagram tried



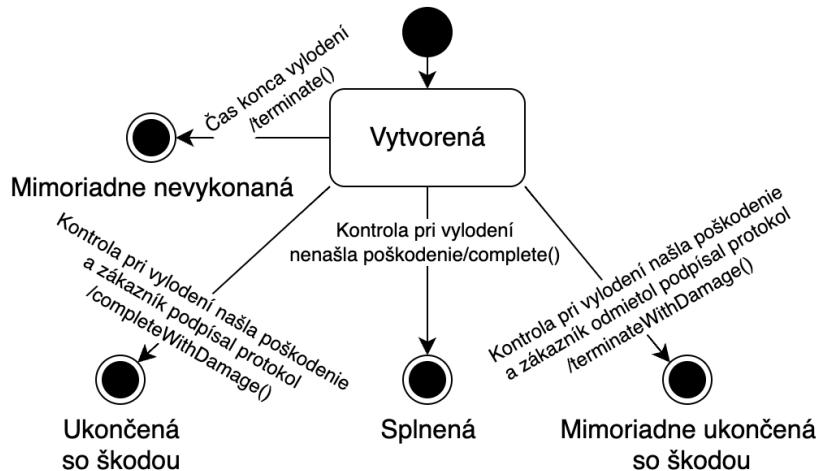
Obr. 4.9: Konceptuálny model: diagram životného cyklu triedy Objednávka



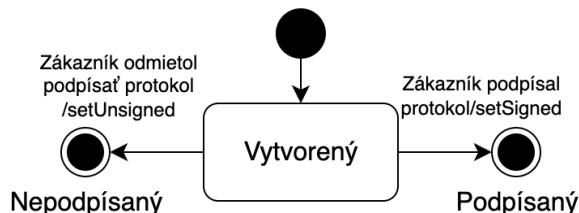
Obr. 4.10: Konceptuálny model: diagram životného cyklu triedy Plavba



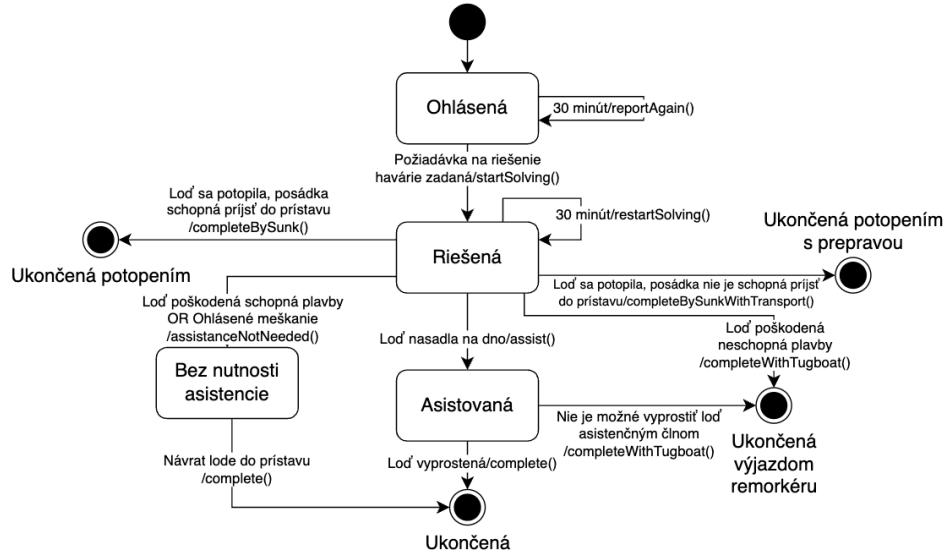
Obr. 4.11: Konceptuálny model: diagram životného cyklu triedy Požiadavka na nalodenie



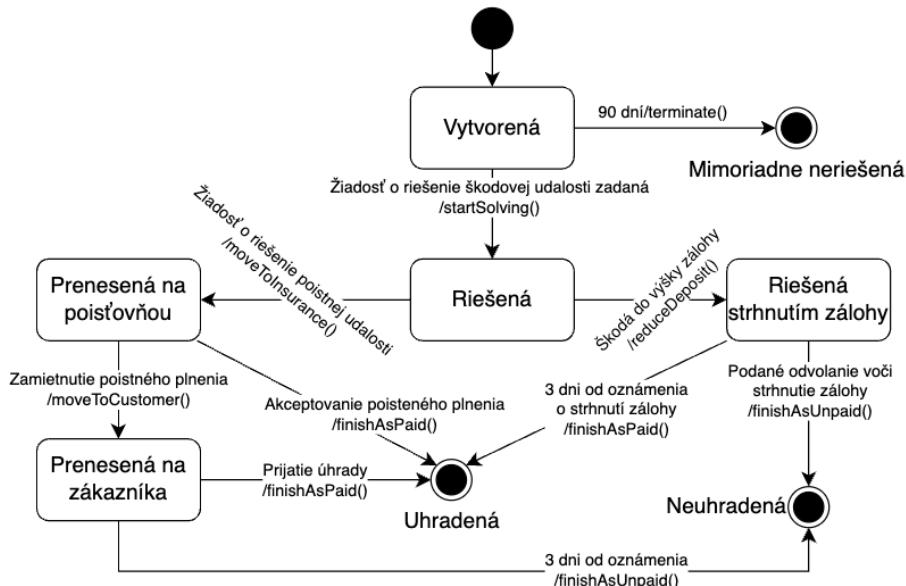
Obr. 4.12: Konceptuálny model: diagram životného cyklu triedy Požiadavka na vylodenie



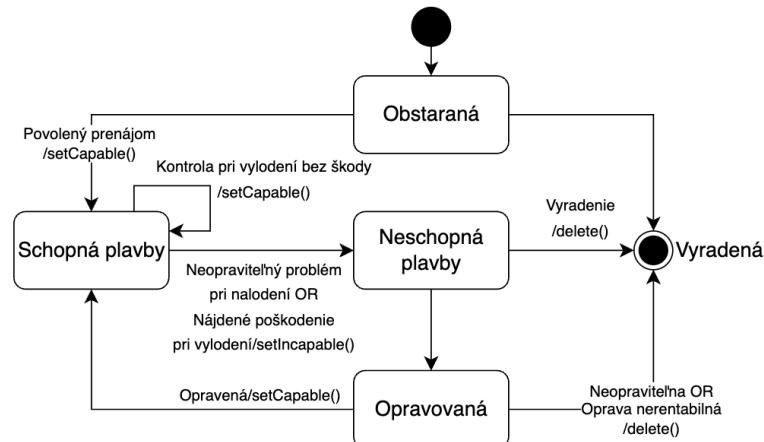
Obr. 4.13: Konceptuálny model: diagram životného cyklu tried Preberací protokol a Odo-vzdávací protokol



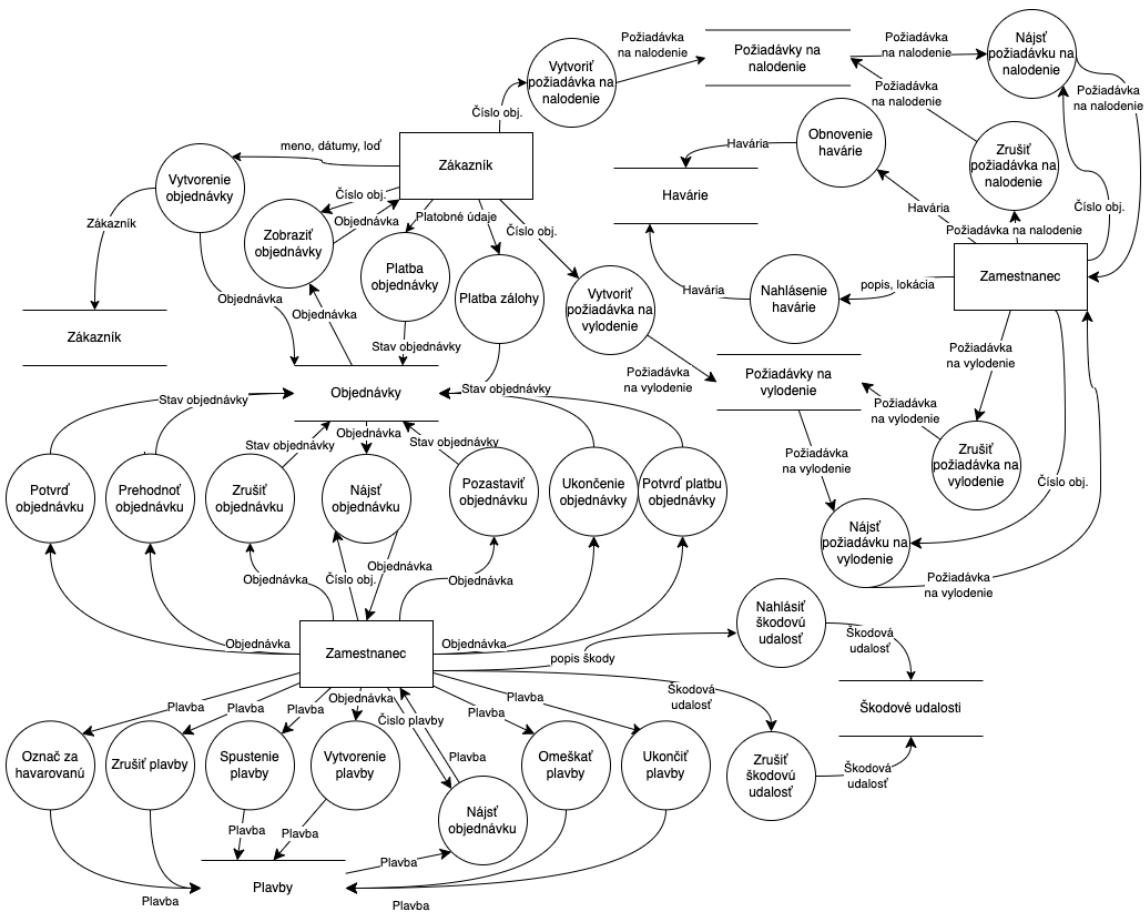
Obr. 4.14: Konceptuálny model: diagram životného cyklu triedy Havária



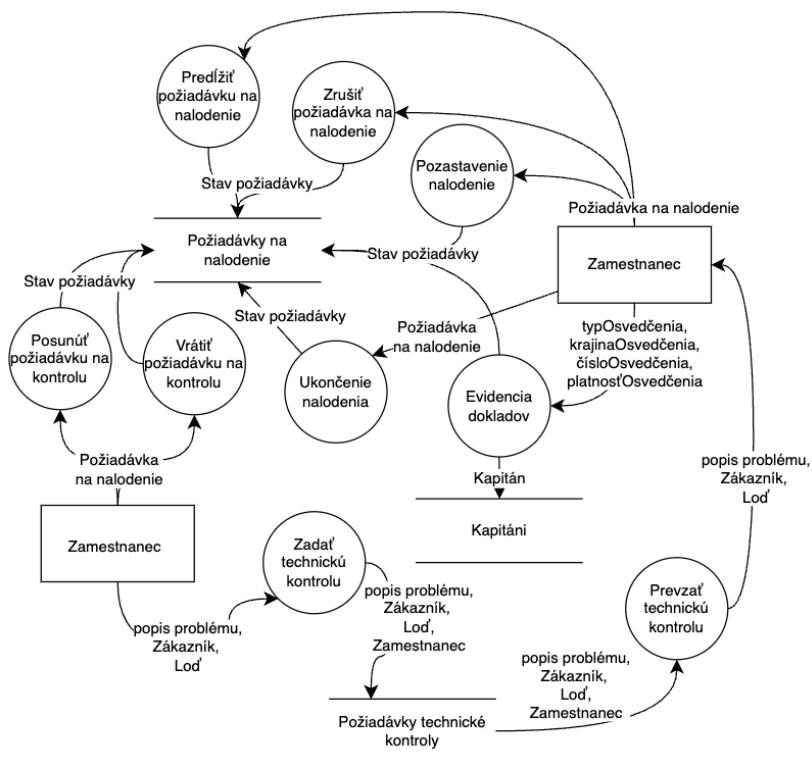
Obr. 4.15: Konceptuálny model: diagram životného cyklu triedy Škodová udalosť



Obr. 4.16: Konceptuálny model: diagram životného cyklu triedy Lod



Obr. 4.17: Konceptuálny model: Diagram dátových tokov kľúčového procesu



Obr. 4.18: Konceptuálny model: Diagram dátových tokov podporného procesu Nalodenie

## 4.2 Technologický model

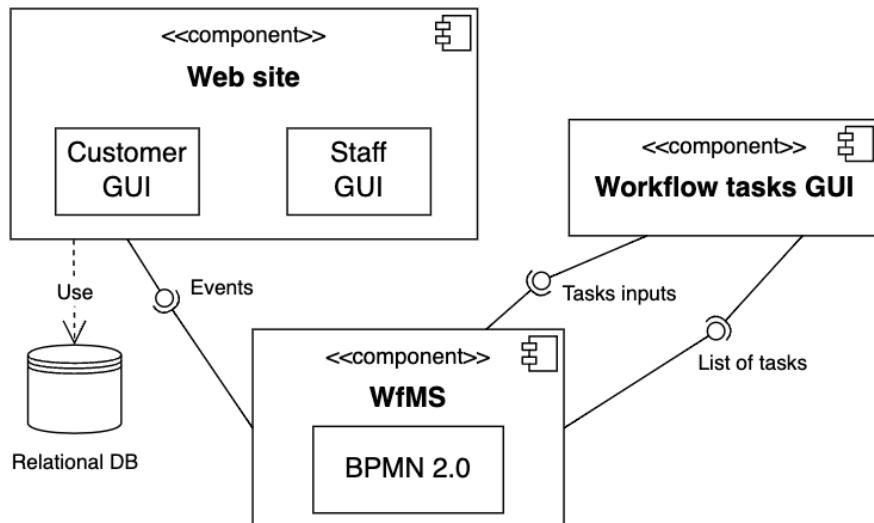
Technologický model tvory podklad pre následný implementačný model. Keďže sa v praxi pri implementácii zvykne používať angličtina, rozhodli sme sa vytvoril technologický model v anglickom jazyku. Anglický jazyk technologického modelu rozširuje možnosti dodávateľa implementácie.

### 4.2.1 Špecifikácia použitej technológie

Autorka tohto kroku Tauchmanová nedefinuje formu takejto špecifikácie. Uvedieme teda UML náčrt (nie diagram), ktorý je doplnený slovným popisom.

Podnik má webovú stránku, prostredníctvom ktorej zákazníci vytvárajú objednávky. Podnik by chcel tento web využiť aj pri zavedení workflow nástroja. Za najlepšiu variantu považuje úplne prepojenie workflow nástroja s touto stránkou. Spokojný bude aj s čiastočným prepojením v podobe vytváranie udalostí cez web a vykonávanie úloh v rozhraní workflow nástroja. Trvá ale natom, aby workflow nástroj podporoval procesný modelovacie štandard BPMN 2.0.

Bude teda zvolený workflow nástroj podporujúci štandard BPMN 2.0. Webová stránka je postavená na servisne orientovanej architektúre. Pribudnú tak služby, ktoré umožnia posielanie udalostí z webového rozhrania na WfMS. Pre komplexnejšiu funkcionalitu vzniknú nové služby. V prípade potreby dátového úložiska sa použije relačná databáza. Ak bude vybraný workflow nástroj ponúkať rozhranie na vykonávanie úloh, bude primárne použité toto rozhranie.



Obr. 4.19: Technological specification sketch

#### 4.2.2 Špecifikácia technických rolí

Z diagramov dátových tokov boli identifikované role zákazníka a zamestnanca. Pri skúmaní detailných procesných diagramov sme vyhodnotili ako vhodný krok rozdelenie role zamestnanca na dve role podľa charakteru činností, ktoré vykonávajú. Zoznam technických rolí vidíme v tabuľke č.4.1.

Role	Description	Intern/Extern	Responsibilities
Customer	Customer	Extern	Creates orders and initiates events related to boarding, landing and incidents.
Admin staff	A worker who performs administrative activities in the office and communicates with the customer.	Intern	Can create events instead of a customer who asks him personally. Solves administrative tasks, incidents and damage events.
Technical staff	A technical worker performs technical activities directly related to the boat.	Intern	Solves reported problems during boarding and controls boat during landing.

Tabuľka 4.1: List of technical roles



#### 4.2.3 Špecifikácia externých udalostí

Name	Description	Role	Alternatives	Multiple
Order created	From web by customer	Customer		yes
Deposit payment received	From web by customer	Customer	• 3 days from order creation; • 7 days from order creation;	no
Payment received	From web by customer	Customer	• 1 month until cruise; • 1 month 1 week until cruise;	no
Customer request boarding on the day of the cruise	From web by customer or admin staff at the customer's request	Customer, Admin staff	End time of boarding on the day of the cruise	yes
Customer requested to board with a new captain	From web by customer or admin staff at the customer's request	Customer, Admin staff	• Customer has requested a postponement of the boarding; • End time of boarding on the day of the cruise;	yes
Customer has requested a postponement of the boarding	From web by customer or admin staff at the customer's request	Customer, Admin staff	• End time of boarding on the day of the cruise; • Customer requested to board with a new captain;	yes
Reporting of an incident	From web by customer or admin staff at the customer's request	Customer, Admin staff	End time of boarding on the day of the cruise	yes
Customer request for landing	From web by customer or admin staff at the customer's request	Customer, Admin staff	Agreed landing time	no

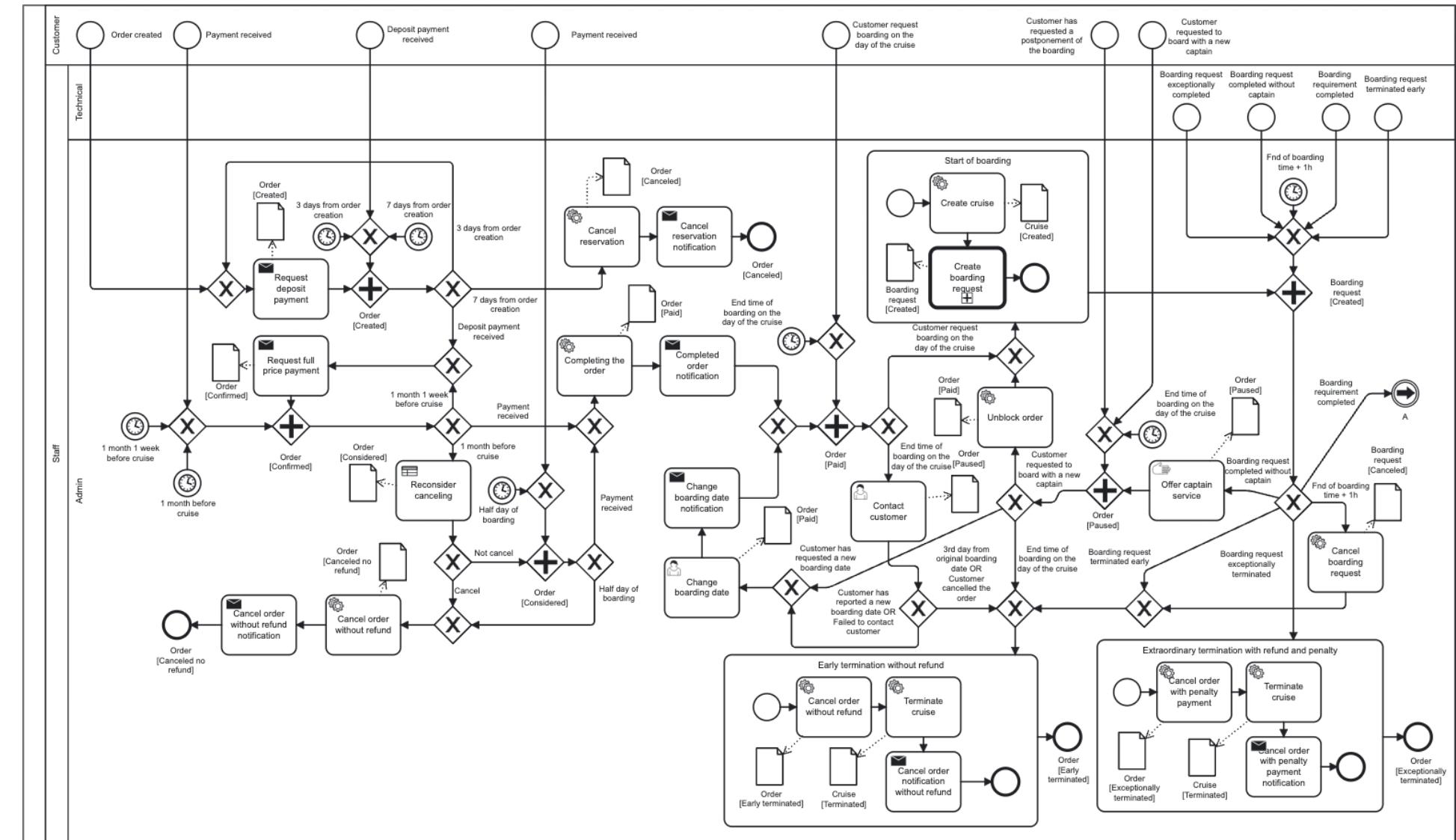
Tabuľka 4.2: List of external events from key process

Name	Description	Role	Alternatives	Multiple
Crew list completed	From web by customer or admin staff at the customer's request	Customer, Admin staff	End time of boarding	no
Reporting end of inspection	From web by customer or admin staff at the customer's request	Customer, Admin staff	End time of boarding	no
Reported problem	From web by customer or admin staff at the customer's request	Customer, Admin staff	• End time of boarding; • Reporting end of inspection;	yes
Reporting end of inspection after end request	From web by customer or admin staff at the customer's request	Customer, Admin staff	15 min	no
Customer signed the acceptance protocol	From task assigned to technical staff	Technical staff	Customer refused to sign acceptance protocol	no
Customer refused to sign acceptance protocol	From task assigned to technical staff	Technical staff	Customer signed the acceptance protocol	no

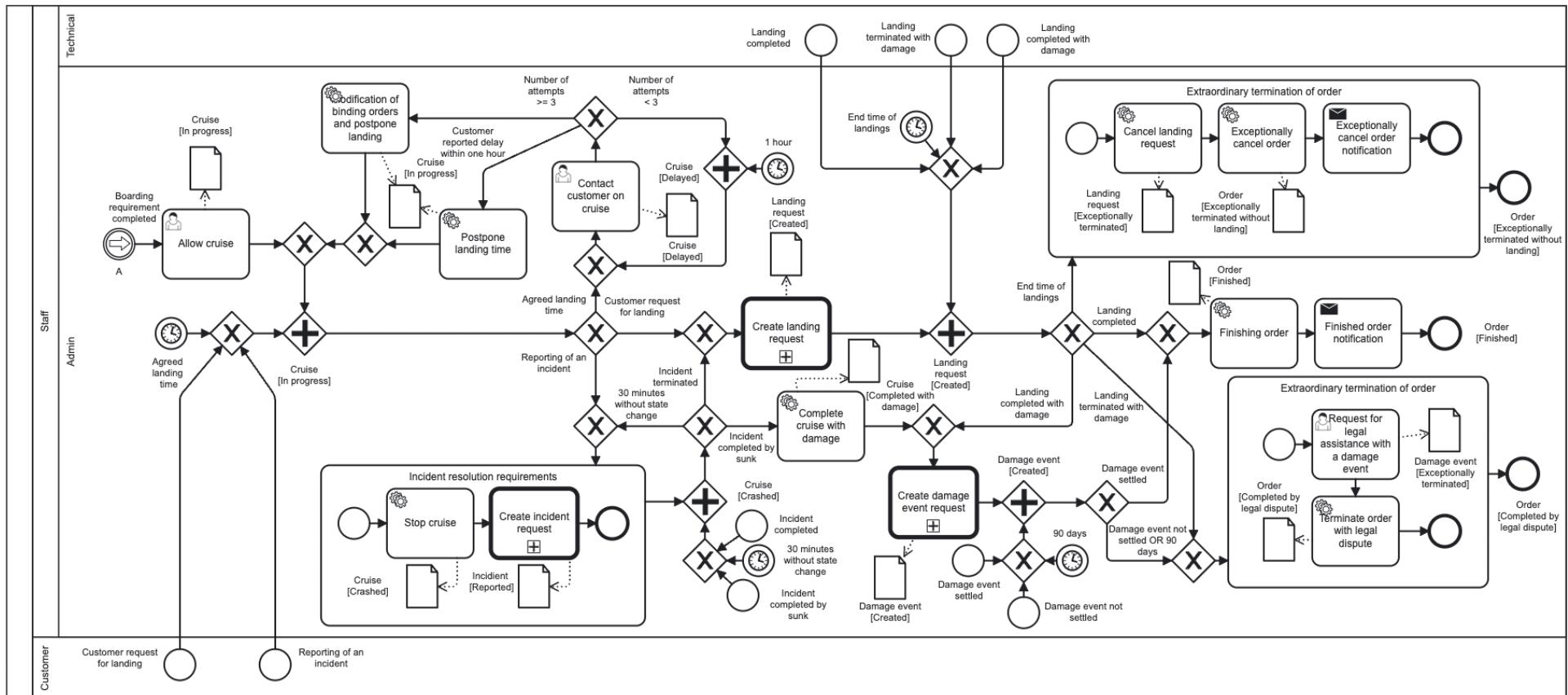
Tabuľka 4.3: List of external events from support process Boarding

#### **4.2.4 Transformácia úloh a pridanie potrebných úloh v závislosti na alternatívnych prechodoch**

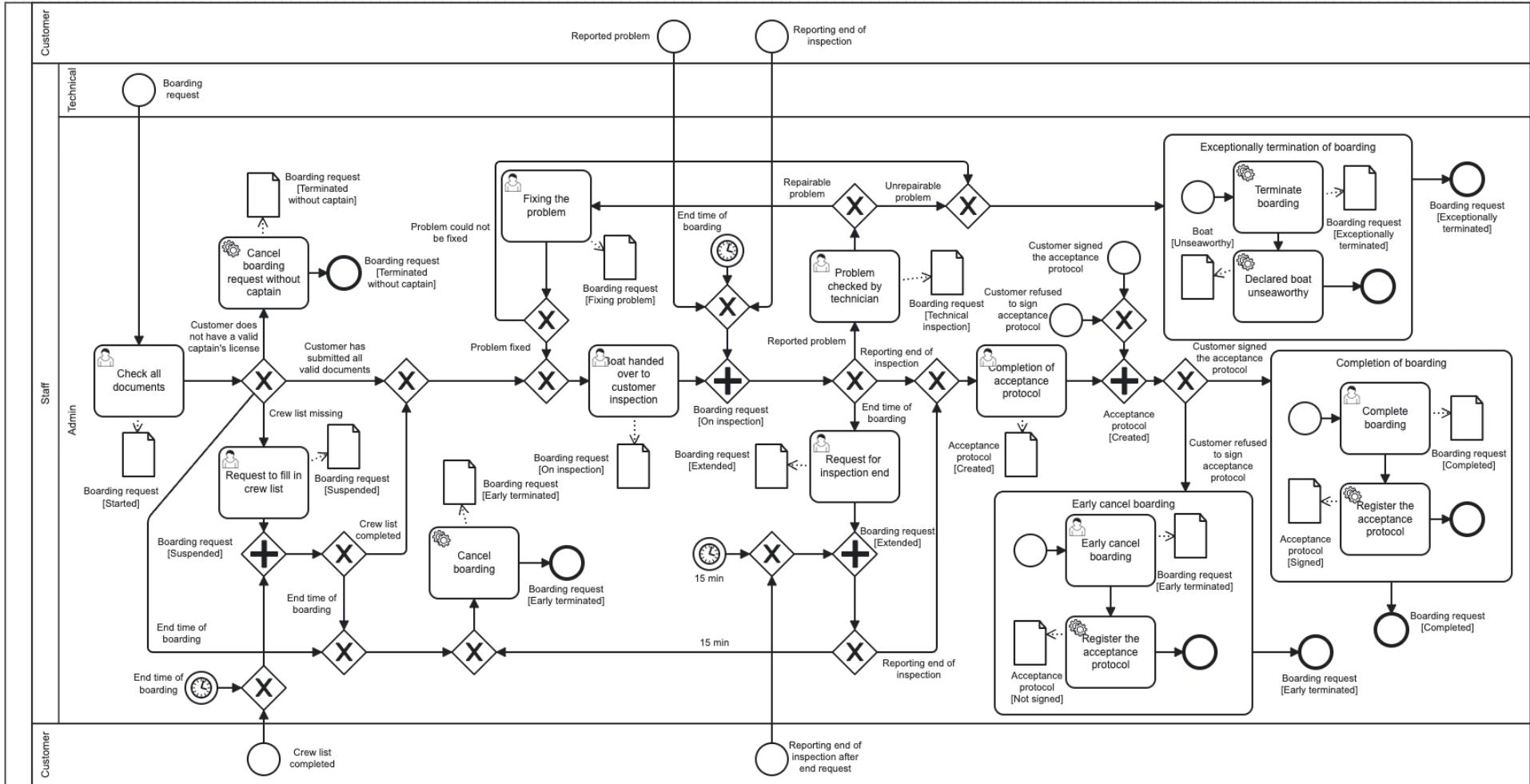
Pri transformovaní úloh sme identifikovali niekoľko rôznych typov úloh. Úlohy, ktoré majú vykonať komplexnejšie operácie sme označili ako *Service task*. Niektoré úlohy sme vyhodnotili potrebu ľudskej interakcie so systémom (*User task*) ale aj fyzickú interakciu zamestnanca so zákazníkom (*Manual task*). Oproti konceptuálnemu modelu pribudlo hlavne niekoľko ďalších notifikácií. Na obrázkoch č.5.57 a č.5.58 vidíme procesný diagram kľúčového procesu po transformovaní a pridaní úloh. Na obrázku č.4.22 vidíme podporný proces nalodenia.



Obr. 4.20: Technological model: process diagram of key process part 1



Obr. 4.21: Technological model: process diagram of key process part 2



Obr. 4.22: Technological model: process diagram of boarding support process



#### 4.2.5 Určenie vstupov a výstupov pre každú úlohu procesu

Task	Inputs	Outputs
Request deposit payment	Customer.email, Order.number, Order.price	Order[Created]
Cancel reservation	Order	Order[Canceled]
Cancel reservation notification	Customer.email, Order.number	
Request full price payment	Customer.email, Order.number, Order.price	Order[Confirmed]
Reconsider canceling	Customer.crediblity, Boat.capacity, Order.cruiseStartAt, seasonStartDate, seasonEndDate	true/false
Cancel order without refund	Customer.email, Order.number	Order[Canceled no refund]
Cancel order without refund notification	Customer.email, Order.number	
Completing the order	Order	Order[Paid]
Completed order notification	Customer.email, Order.number	
Contact customer	Customer.name, Customer.phoneNumber, Customer.email, Order.number, Order.cruiseStartAt, Boat.type, Boat.name	Order[Paused]
Change boarding date	Order, maxBoarPostponeDays, newBoardingDate	Order[Paid]
Change boarding date notification	Customer.email, Order.number, Order.cruiseStartAt	
Cancel order without refund	Order	Order[Early terminated]
Terminate cruise		Cruise[Terminated]
Cancel order without refund notification	Customer.email, Order.number	
Unblock order	Order	Order[Paid]
Create cruise	Order	Cruise[Created]
Create boarding request	Cruise	Boarding request[Created]

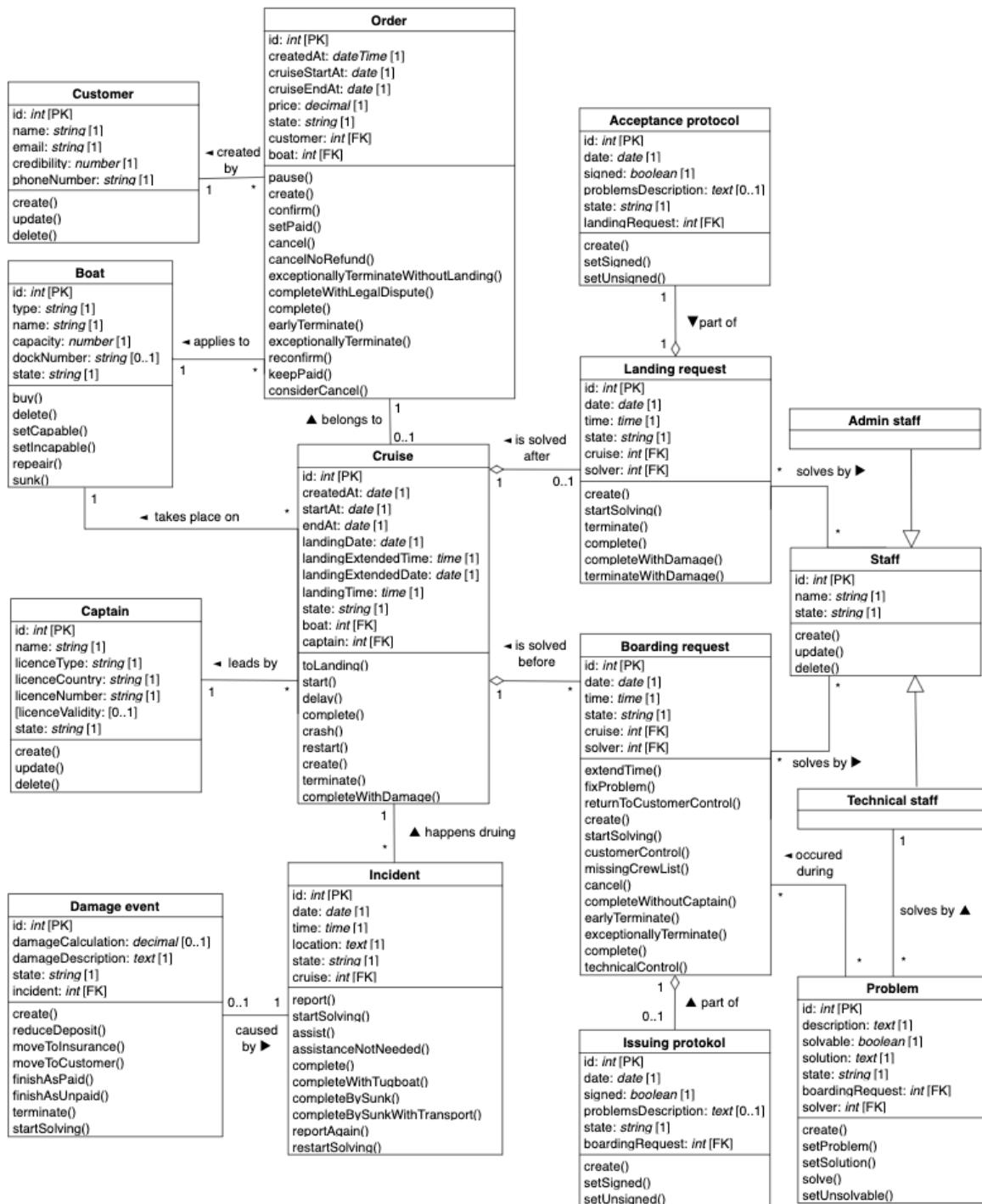
Offer captain service	Order	Order[Paused]
Cancel boarding request	Boarding request	Boarding request[Canceled]
Cancel order with penalty payment	Order, boarRefundPercentage	Order[Exceptionally terminated]
Cancel order with penalty payment notification	Customer.email, Order.number	
Allow cruise	Cruise	Cruise[In progress]
Modification of binding orders and postpone landing	Cruise, landPostponeDays	Cruise[In progress]
Postpone landing time	Cruise, landPostponeTime	Cruise[In progress]
Contact customer on cruise	Cruise Customer maxPhoneAttempts	Cruise[Delayed]
Stop cruise	Cruise	Cruise[Crashed]
Create incident request	Cruise	Incident[Reported]
Complete cruise with damage	Cruise	Cruise[Completed with damage]
Create landing request	Cruise	Landing request[Created]
Create damage event request	Cruise	Damage event[Created]
Cancel landing request	Landing request	Landing request[Exceptionally terminated]
Exceptionally cancel order	Order	Order [Exceptionally terminated without landing]
Exceptionally cancel order notification	Customer.email, Order.number	
Finishing order	Order	Order[Finished]
Finished order notification	Customer.email, Order.number	
Request for legal assistance with a damage event		Damage event[Exceptionally terminated]
Terminate order with legal dispute	Order	Order[Completed by legal dispute]

Tabuľka 4.4: List of inputs and output for tasks in key process

Check all documents	Boarding request	Boarding request[Started]
Cancel boarding request without captain	Boarding request	Boarding request[Terminated without captain]
Request to fill in crew list	Boarding request	Boarding request[Suspended]
Cancel boarding	Boarding request	Boarding request[Early terminated]
Boat handed over to customer inspection	Boarding request	Boarding request[On inspection]
Problem checked by technician	Customer.name, Boat.type Boat.name Boat.dockNumber description	Boarding request[Technical inspection], solvable, description
Fixing the problem	Boarding request, Boat.type Boat.name Boat.dockNumber description, solvable	Boarding request[Fixing problem] Problem[Created] Problem(solvable, description, solution)
Terminate boarding	Boarding request	Boarding request [Exceptionally terminated]
Declared boat unseaworthy	Boat	Boat[Unseaworthy]
Request for inspection end	Boarding request	Boarding request[Extended]
Completion of acceptance protocol	Boarding request	Acceptance protocol [Created]
Early cancel boarding		Acceptance protocol [Not signed]
Register acceptance protocol	Boarding request, signed	Acceptance protocol[Signed/ Not signed] Acceptance protocol(date, signed)
Complete boarding	Boarding request	Boarding request[Completed]

Tabulka 4.5: List of inputs and output for tasks in support process Boarding

#### 4.2.6 Aktualizácia diagramu tried



Obr. 4.23: Technological model: class diagram

#### 4.2.7 Špecifikácia notifikácií

Task	Receiver	Sender	Subject
Request deposit payment	{Customer.email}	info@charter.com	Deposit required

**Text:**  
 We remind you that  
 the deposit for the order n.{Order.number}  
 must be paid before the 7th day from order creation.  
 The Charter company

Tabuľka 4.6: Notification for deposit payment

Task	Receiver	Sender	Subject
Request full price payment	{Customer.email}	info@charter.com	Full price payment required

**Text:**  
 We remind you that  
 full price ({Order.price}) for the order n.{Order.number}  
 must be paid before the 7th day from order creation.  
 The Charter company

Tabuľka 4.7: Notification for full price payment

Task	Receiver	Sender	Subject
Cancel reservation notification	{Customer.email}	info@charter.com	Cancel reservation

**Text:**  
 Unfortunately, I have to inform you  
 that we canceled your order n.{Order.number}  
 because you did not pay the deposit.  
 The Charter company

Tabuľka 4.8: Notification for cancel reservation

Task	Receiver	Sender	Subject
Cancel order without refund notification	{Customer.email}	info@charter.com	Cancel order

**Text:**

Unfortunately, I have to inform you  
that we canceled your order n.{Order.number} without refundation.  
For further information, contact  
the customer department at +421999111222.

The Charter company

Tabulka 4.9: Notification for cancel reservation without refund

Task	Receiver	Sender	Subject
Completed order notification	{Customer.email}	info@charter.com	Completed order

**Text:**

Your order n.{Order.number} has been paid and completed.  
We look forward to your arrival on {Order.cruiseStartAt}  
Arrival for boarding is required from 8:00 to 16:00.

The Charter company

Tabulka 4.10: Notification for completed order

Task	Receiver	Sender	Subject
Change boarding date notification	{Customer.email}	info@charter.com	Boarding date changed

**Text:**

Boarding date for order n.{Order.number}  
has been changed to {Order.movedBoardingDate}.  
Arrival for boarding is required from 8:00 to 16:00.

The Charter company

Tabulka 4.11: Notification for changed boarding date

Task	Receiver	Sender	Subject
Cancel order with penalty payment notification	{Customer.email}	info@charter.com	Order exceptionally cancelled

**Text:**

We are sorry, but your order n.{Order.number} has been exceptionally cancelled.  
We apologize for the caused problems. You will be refunded full price plus 20% of the rental price and you are entitled to adequate financial compensation according to the General Business Terms.

The Charter company

Tabulka 4.12: Notification for cancel order with penalty

Task	Receiver	Sender	Subject
Exceptionally cancel order notification	{Customer.email}	info@charter.com	Landing exceptionally unrealized

**Text:**

We apologize that we could not make the planned landing.  
You will be contacted by phone regarding further information.  
We apologize for the caused problems. You will be refunded full price plus 25% of the rental price and you are entitled to adequate financial compensation according to the General Business Terms.

The Charter company

Tabulka 4.13: Notification for exceptionally canceled order

Task	Receiver	Sender	Subject
Finished order notification	{Customer.email}	info@charter.com	Boarding date changed

**Text:**

Your rental (order n.{Order.number}) has been successfully completed.  
We look forward to your next cruise on one of our ships.

The Charter company

Tabulka 4.14: Notification for finished order

## **4.3 Zhodnotenie a doplnenie metodického postupu transformácie do technologického modelu**

V predošej sekcii sme použitím metodického postupu overili jeho zrozumiteľnosť a výkonateľnosť. Vzniknutá architektúra v takejto podobe by mala byť úplná. Identifikovali sme ale podmienky, za ktorých architektúra nie je úplná. Autori metodického postupu pri vytváraní procesov nepoužili komplexné rozhodnutia riadiace sa business pravidlami, ktoré v majú v procesnom diagrame podobu *Business Rule Task*, preto do postupu nezahrnuli kroky, ktoré s tým súvisia. Pri prepájaní viacerých systémov, ktoré si medzi sebou vymieňajú dátá je ale automatizácia komplexných rozhodnutí veľkým prínosom. Napriek tomu, že komplexné business pravidlá môžu byť náročnou tému, je nevhodné opomenúť túto tému a je potrebné jú zohľadniť v metodickom postupe. V sekcii 4.3.1 tak navrhujeme **rozšírenie metodického postupu** o nepovinný krok, vykonávaný pre **podporu business pravidiel pri komplexných business rozhodnutiach**.

Architektúra vytvorená pomocou metodického postupu má ale jeden ďalší drobný nedostatok. Nezohľadňuje jednu veľmi dôležitú vlastnosť architektúry, ktorou je **udržateľnosť** a hľavne aj jej pridruženú vlastnosť (Kozolek 2011) **modifikovateľnosť**. Preto v sekcii 4.3.2 navrhuje **rozšírenie metodického postupu o zoznam business premenných**. Teda hodnoty, ktoré sú dôležité pre business a majú tendenciu meniť sa časom alebo business rozhodnutím. V dôsledku pridania tohto kroku sme v sekcii 4.3.3 navrhli doplnenie už existujúcich krovov.

#### 4.3.1 Pridanie rozhodovacích modelov a tabuľiek

##### *Umiestnenie*

Tento krok **nie je povinný** a vykonáva sa len vtedy ak dochádza v procese ku vyhodnocovaniu zložitejších business pravidiel. Mal by nasledovať minimálne po kroku číslo 4. *Transformácia a pridanie úloh*, pretože už sú identifikované úlohy zodpovedné za business rozhodnutia. Ale vzhľadom nato, ako komplexné môžu byť business rozhodnutia, **doporučujeme ho vykonávať až ako predposledný bod** celého metodického postupu, pred krokom *Pridanie ďalšieho popisu či poznámok*.

##### *Čo?*

Vytvorenie grafického zobrazenia komplexných business rozhodnutí pomocou grafu pomocou štandardnej DMN notácie ([Decision Model and Notation 2023](#)), ktorá definuje graf rozhodovacích požiadaviek (Decision Requirements Graph, DRG) a jeho vizuálnu reprezentáciu v podobe diagramu rozhodovacích požiadaviek (Decision Requirements Diagram, DRD) pre každú úlohu typu *Business Rule Task*. Podľa DMN pravidiel vytvoriť pre každé rozhodnutie v DRD rozhodovaciu tabuľku.

##### *Prečo?*

V procesoch môže často dochádza ku potrebe urobiť business ([Decision Model and Notation 2023](#), std. 11), rozhodnutie, ktoré má v podniku jasne stanovené pravidlá (napríklad schválenie úveru). K takýmto rozhodnutiam sú potrebné nejaké vstupy, ktoré môžu byť staticky zadané ale aj dynamicky dotahované či vyhodnocované v inom systéme. Jedno business rozhodnutie sa môže skladať z viacerých menších rozhodnutí a komplexita tak výrazne narastá. Preto je potrebné mať prehľad o vstupoch, rozhodnutiach a ich vzájomných závislostiach.

##### *Ako?*

1. identifikovať vstupy, výstupy a zdroj znalostí pre business rozhodnutie Je potrebné prejsť všetky úlohy typu *Business Rule Task* a brány, ktoré sa nachádzajú bezprostredne za nimi. Pri takýchto úlohách by sa mal nachádzať doplňujúci text, ktorý v krátkosti vysvetluje business rozhodnutie. Brána a cesty, ktoré z úlohy vychádzajú by mali popisovať podobu výstupu rozhodnutia. Ak doplňujúce texty chýbajú, je potrebné konzultovať business rozhodnutia s vlastníkom procesu. Z textov je treba identifikovať aký výstup má business rozhodnutie, aké vstupy ho ovplyvňujú a prípadne aj zdroj znalostí pre business rozhodnutie. Pre lepšie pochopenie tohto kroku je vhodné prečítanie kapitoly Základne koncepty v špecifikácií DMN ([Decision Model and Notation 2023](#), str. 11–16).

2. Vytvoriť DRD pre všetky identifikované business rozhodnutia Informácie získane z predošlého bodu je potrebné zaznamenať graficky v podobe DRD diagramu. Pre lepšie pochopenie tohto kroku je vhodné prečítanie kapitoly Požiadavky (DRG a DRD) v špecifikácii DMN (Decision Model and Notation 2023, str. 17–46).
3. Vytvoriť prvý návrh rozhodovacích tabuľiek pre každé rozhodnutie z DRD diagramov "Rozhodovacia tabuľka je tabuľková reprezentácia množiny súvisiacich vstupných a výstupných výrazov usporiadaných do pravidiel označujúcich, ktoré výstup sa vzťahuje na konkrétnu množinu výstupu. Pri jednoduchších rozhodnutiach môže byť tvorba tejto tabuľky relatívne jednoduchá. Treba si dať hľavne pozor nato, že pri práci s množinou vstupov nesmieme zabudnúť na hraničné hodnoty množín. Pre lepšie pochopenie tohto kroku je vhodné prečítanie kapitoly Rozhodovacia tabuľka v špecifikácii DMN (Decision Model and Notation 2023, str. 59–77). Pri prvom návrhu tabuľky nie je nutné používať v podmienkach jazyk FEEL úplne korektne. Dôležité je mať dostatočný podklad, ktorý vieme prediskuskutovať s vlastníkom procesu.
4. Diskutovať štruktúru business rozhodnutí s vlastníkom procesu S vlastníkom procesu si treba vyjasniť správnosť a úplnosť business rozhodnutí vyjadrených pomocou DRD a rozhodovacích tabuľiek. Ďalej je potrebné overiť správnosť rozhodovacích tabuľiek, prípadne získať zdroj znalostí, na základe ktorého vieme správnosť tabuľky vyhodnotiť. Pri komplexných pravidlách je vhodné do tabuľky pridať textový komentár pred ďalší krok.
5. Finalizovať rozhodovacie tabuľky Po diskusii s vlastníkom procesu musíme byť schopný prerobiť rozhodovacie tabuľky do takmer finálnej podoby. Predovšetkým je potrebné skontrolovať pravidlá a upraviť ich do správnej formy jazyka FEEL (Decision Model and Notation 2023, str. 79–162).

### 4.3.2 Špecifikácia business premenných

#### *Umiestnenie*

Tento krok je vhodné vykonať za krokom číslo 5. *Určenie vstupov a výstupov pre každú úlohu procesu*, pretože je z neho možné čerpať niektoré informácie. Zároveň je vhodné ho vykonávať pred krokom č.6 *Aktualizácia diagramu tried* aby sa zabránilo prípadnému chybnému pridaniu atribútov objektom.

#### *Čo?*

Zoznam business premenných a konštánt s ich umiestnením a účelom. Ide o hodnoty, ktoré sú pre business dôležité, ovplyvňujú podnikový proces ale nie sú atribútom objektov. Majú tendenciu meniť sa časom alebo nejakým business rozhodnutím. Hodnota je definovaná business logikou alebo externými faktormi (zákony) a nie technickým spracovaním (pomocné premenné). V procese sa obvykle používajú pri podmienkach v jednoduchých (XOR brány) či komplexných (Business Rule Task) rozhodovaniach. Môžu vstupovať do nejakých výpočtov alebo sa iba staticky zobrazovať na viacerých miestach (formulároch, emailoch, a pod.).

#### *Prečo?*

Ak technologický model neobsahuje takéto premenné a konštanty, je znížená udržateľnosť a modifikovateľnosť architektúry z ktorej vyplýva viacero potenciálneho problémov, ktoré sa objavia až pri zmenových požiadavkách. Pri staticky definovaných hodnotách môže nastať problém pri zmene hodnoty, ak je hodnota používaná a má byť zmenená na viacerých miestach. Môže ale nastať aj opačný problém kedy je hodnota v procese používaná pomocou premennej na viacerých miestach. Tie ale nemusia mať rovnaký business význam a zhoda hodnôt môže byť len dočasná. Pri zmene hodnoty premennej tak dôjde k zmene aj na významovo odlišných miestach a tento problém prinajlepšom odhalí až testovanie. V oboch prípadoch je zvolený zlý spôsob implementácie na základe nedostatku informácií o danej hodnote v technologickom modeli. Pri implementácii je potrebné vedieť, ktoré hodnoty je vhodné používať staticky či dynamicky, a ktoré majú potenciál byť časom zmenené. Pri zmenových požiadavkách budeme vedieť lepšie odhadnúť ako je systém pripravený na zmene a aké potenciálne problémy môžu zmenou vzniknúť. Často meniace sa premenné môžeme do systému vkladať pomocou konfiguračných premenných, čo môže znížiť potrebu upravovať diagramy či zdrojový kód a následne nasadzovať tieto úpravy.

#### *Ako?*

1. preskúmanie časových udalostí v procesnom diagrame Väčšina časových udalostí v procese vyplýva z business logiky alebo externých netechnických faktorov. V konceptuálnom modeli môžeme naraziť na explicitne uvedené časové hodnoty (napr. 24 mesiacov) alebo implicitne uvedené časové hodnoty (napr. po skončení záruk). Každú explicitne uvedenú hodnotu by sme mali byť schopní vyjadriť aj slovne v

kontexte jej použitia (napr. záručná doba). A naopak pre každú implicitnú hodnotu potrebujeme vedieť vyjadriť číselne. Pri špecifikovaní business premenných sa snažíme nájsť slovné pomenovanie číselnej hodnoty. Ak natrafíme na časovú udalosť, ktorá je vyjadrená slovne a neobsahuje žiadne číselné vyjadrenie tak hodnotu tak číselnú hodnotu je nutné vypočítať. Tu môže ísť o výpočet do ktorého vstupuje viacero business premenných (napr. koniec pracovného dňa) alebo business premenná a atribút objektu (napr. mesiac od vytvorenia objednávky).

2. vytvorenie zoznamu business premenných Po správnom pomenovaní časových udalostí

vytvoríme tabuľku business premenných v ktorej uvádzame všetky relevantné informácie o premennej. Minimálne informácie, ktoré by sme mali evidovať vidíme vo vzorovej tabuľke č.4.17. Ak je hodnota premennej tvorená výpočtom, je potrebné tento výpočet uviesť jednoznačným spôsobom. Vhodným vyjadrovacím nástrojom je jazyk FEEL. Môžeme ale namiesto FEEL alebo matematického výrazu uviesť slovný komentár popisujúci výpočet.

3. vyhľadať business premenné v zozname vstupov a výstupov úloh V zozname vstup a

výstupov by sa mali nachádzať všetky ďalšie premenné, ktoré majú potenciál byť business premennými. Musíme zvážiť ktoré vstupy a výstupy nepatriace objektom majú čisto technický charakter (napr. počítadlo prechodu úlohou) a ktoré majú business charakter (napr. maximálny počet prechodov cez úlohu). Môže sa jednat o nové business premenné alebo len prepoužitie už existujúcej, ktorá napríklad vstupovala do výpočtov pri časových udalostiach. Nové nájdene premenne doplníme do zoznamu business premenných, pri už existujúcich premených doplníme ďalší výskyt.

4. diskutovať zoznam business premenných s vlastníkom procesu Je potrebné si vyjasniť správnosť a úplnosť zoznamu s vlastníkom procesu, prípadne doplniť informácie, ktoré neboli jasné z doterajších modelov. Zároveň je vhodné prediskutovať aj možnú frekvenciu zmien hodnoty premených a poznačiť to do poznámok.

5. nahradiť hodnoty časových udalostí názvom premennej Po diskusii s vlastníkom

procesu by mal byť zoznam business premenných kompletný. Môžeme tak nahradiť explicitné hodnoty časových udalostí v technologickom diagrame za názvy business premenných.

Názov	Hodnota	Využitie	Umiestnenie	Zdroj	Poznámka
záručná doba	24 mesiacov	<ul style="list-style-type: none"> <li>- časová udalosť</li> <li>- časová udalosť</li> <li>- notifikácia</li> </ul>	<ul style="list-style-type: none"> <li>- klúčový proces v stave Objednávka[Doručená]</li> <li>- podporný proces Obstaranie tovaru v stave Naskladnenie[Ukončené]</li> <li>- úloha Notifikácia o doručení objednávky;</li> </ul>	Spotrebiteľský zákon č.XY	
doba refundácie	7 dní	<ul style="list-style-type: none"> <li>- podmienka</li> <li>- notifikácia</li> </ul>	<ul style="list-style-type: none"> <li>- XOR za úlohou Zrušenie rezervácie</li> <li>- úloha Notifikácia o vytvorení rezervácie</li> </ul>	Všeobecné obchodné podmienky podniku	často menené
vernostná zľava	12%	- vstup do úlohy	- úloha Výpočet ceny	Vedúci PR oddelenia	

Tabuľka 4.15: Zoznam business premenných

### 4.3.3 Špecifikácia notifikácií

Ako? (Pôvodný postup)

1. identifikovať úlohy, v ktorých sú notifikácie poslané,
2. diskutovať znenie textov notifikácií, vrátane odosielateľov a príjemcov, s vlastníkom procesov, prípadne s kľúčovými užívateľmi,
3. vytvoriť tabuľku pre každú notifikáciu, ktorá obsahuje potrebné informácie na základe charakteru notifikácie.

V dôsledku predošej navrhovanej zmeny *Špecifikácia business premenných* je potrebné doplniť následujúci bod.

4. identifikovať výskyt business premenných v notifikáciách a korektne ich upraviť v tabuľke notifikácií

V notifikácií sa môžu a často aj objavujú business premenné. Tie treba vyhľadať a korektne ich v tabuľke notifikácií nahradiať v podobe premennej. Je vhodné naznačiť, že sa jedná o business premennú napríklad takýmto spôsobom {B:NázovPremennej}. Pri implementácii tak bude jasné odkiaľ sa bude hodnota čerpať a bude zvolený lepší spôsob implementácie.

Tauchmanová sice aktualizovala podobu tabuľky notifikácií. Tá ale nezohľadňuje výskyt špeciálnych hodnôt ako sú odkazy či súbory na stiahnutie. Niektoré takéto hodnoty môžu byť zároveň business premennou (napr. emailová adresa), iné môžu byť čisto statické (odkaz na Wikipediu). Prínosy evidencie takýchto špeciálnych premenných sú podobne ako pri zavedení business premenných. Ide predovšetkým o zvýšenie udržateľnosti a modifikovateľnosti celej architektúry.

5. identifikovať výskyt špeciálnych premenných v notifikáciách a korektne ich upraviť v tabuľke notifikácií

V notifikácií sa môžu objavovať technické premenné, ktoré môžu mať podobu odkazov alebo súboru na stiahnutie. Hodnota týchto premenných môže byť statická alebo môže mať podobu business premennej. Je potrebné identifikovať takéto premenné a vhodné naznačiť, že sa jedná o inú ako objektovú premennú. Vhodný spôsob je {S:NázovPremennej}. Následne doplníme tabuľku notifikácií o relevantné informácie. Ukážku vidíme v tabuľke č.4.16.

Úloha	Príjemca	Odosielateľ	Predmet
		{B:podnikovyEmail}	
<b>Text:</b>			
Doporučujeme si prečítať {S:vos}.			
Špeciálne pre-menné	Text	Typ	Zdroj/Hodnota
vos	"Všeobecné Obchodné Podmienky"	súbor na stiahnutie	Generálny sekretár
staznosti	"webového formuláru"	odkaz na web	IT oddelenie / "http:example.com"

Tabuľka 4.16: Rozšírená tabuľka notifikácií s vyznačenými špeciálnymi premennými

## 4.4 Doplnenie technologického modelu

V tejto sekcii doplníme technologický model vytvorený pomocou metodického postupu od autorov Pavelčák a Tauchmanová o nové kroky, ktoré sme navrhli v predošej sekcii č.4.3.

### 4.4.1 Špecifikácia business premenných

Pri vykonávaní prvého kroku *preskúmanie časových udalostí v procesnom diagrame* sme vytvorili tabuľku s takmer dvadsať business premennými. Niektoré z nich majú v sebe priamo hodnotu (napr. 16:00), niektoré prepoužívajú hodnotu iných a dopĺňajú ju o ďalšiu business logiku (napr. {boardingEndTime} at current day).

Následne sme v zozname vstupov a výstupov úloh identifikovali ďalších 6 business premenných:

- seasonStartDate
- seasonEndDate
- maxBoardPostponeDay
- boardRefundPercentage
- landPostponeDays
- landPostponeTime

V tabuľke 4.17 vidíme všetky identifikované business premenné. V poslednej časti tabuľky vidíme aj premenné, ktoré boli identifikované až neskôr po špecifikácii notifikácií. Následne sme v technologickom procesnom diagrame nahradili explicitne uvedené časové hodnoty za názvy business premenných. Na obrázkoch č.4.24 a č.4.25 vidíme finálnu podobu technologického procesného diagramu klíčového proces. V diagrame podporného procesu *boarding* sa nachádza jedna business premenná. V zozname business premenných je ale

uvedené, že vychádza zo všeobecných obchodných podmienok a v poznámke je uvedené, že sa nebude hodnota meniť. Preto už nebolo potrebné diagram upravovať a finálna veria je teda na obrázku 4.22.

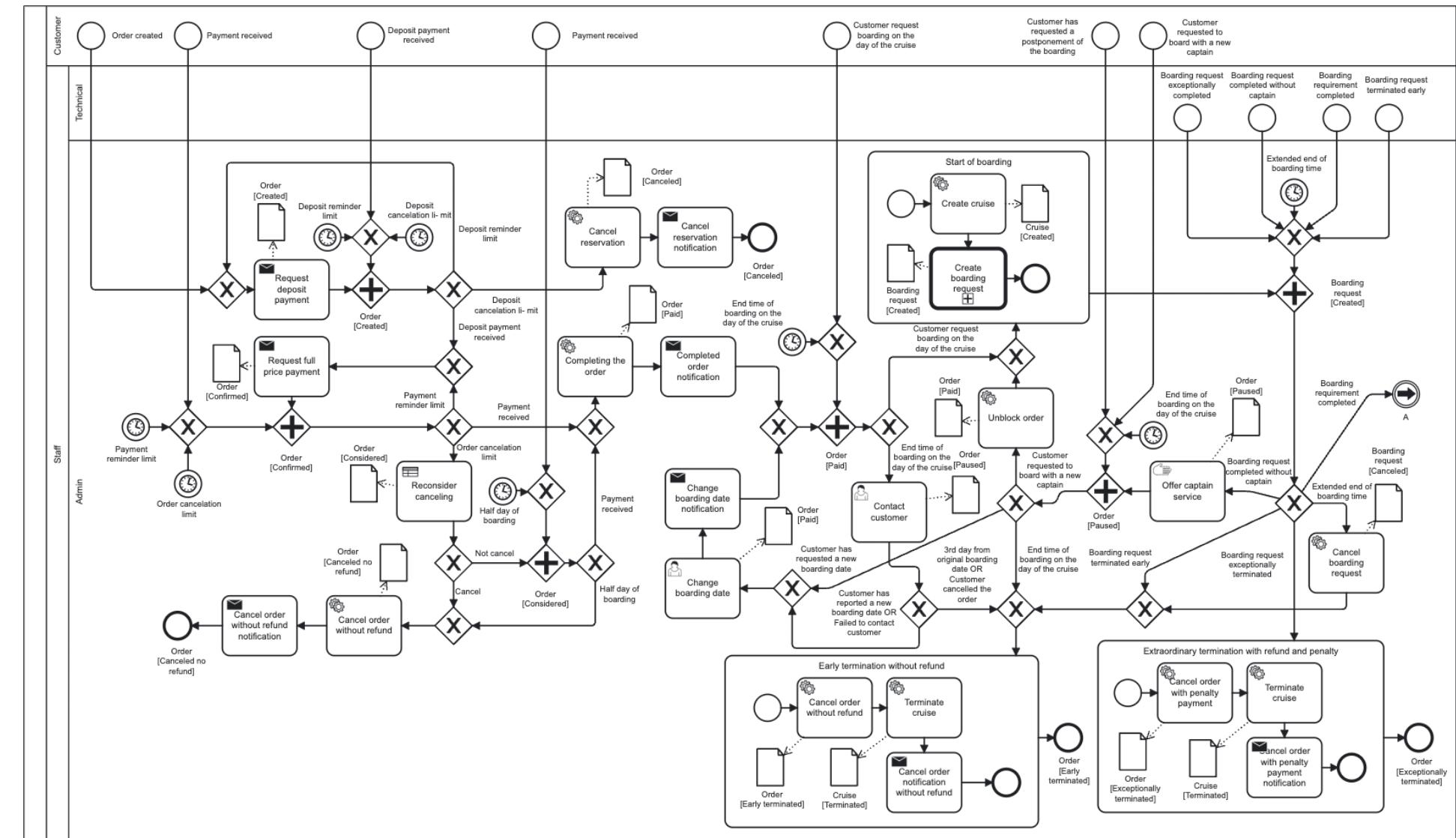
Name	Value	Usage	Location	Source	Notes
Deposit reminder limit	3 days from {Order.createdAd}	- time event - notification	- key process state Order[Created] - task Request deposit payment	General business terms: Payment conditions	
Deposit cancellation limit	7 days from {Order.createdAd}	- time event - notification	- key process state Order[Created] - task Request full price payment	General business terms: Payment conditions	
Payment reminder limit	1 month 1 week before {Order.cruiseStartAt}	- time event	- key process state Order[Confirmed]	General business terms: Order cancellation	
Order cancellation limit	1 month before {Order.cruiseStartAt}	- time event	- key process state Order[Confirmed]	General business terms: Order cancellation	
Half day of boarding	*notes describing the calculation	- time event	- key process state Order[Considered]	General business terms: Order cancellation	time between business variables {boardingStartTime} and {boardingStartTime}
boardingStartTime	8:00	- calculation - notification - notification	- business variable Half day of boarding - task Completed order notification - task Change boarding date notification	Head of the technical department	changes twice a year

	boardingEndTime	16:00	- calculation - notification - notification	- business variable Half day of boarding - task Completed order notification - task Change boarding date notification	Head of the technical department	changes twice a year
	End time of boarding on the day of the cruise	time {boardingStartTime} at date {Order.cruiseStartAt}	- time event	- key process state Order[Paid]	Head of the technical department and General business terms	
	Extended end of boarding time	{boardingEndTime} + {extendBoardingHours}	- time event	- key process state Boarding request[Created]	General business terms: Boarding	
<i>Name</i>		<i>Value</i>	<i>Usage</i>	<i>Location</i>	<i>Source</i>	<i>Notes</i>
extendBoardingHours		1 hour	- calculation	- business variable Extended end of boarding time	Head of the technical department and General business terms	it is reviewed annually
Agreed landing time		time {landingStartTime} or {Cruise.landingExtendedTime} at date {Cruise.landingDate} or {Cruise.landingExtendedDate}	- time event	- key process state Cruise[In progress]	Head of the technical department and General business terms: Landing	extended time and date is used when Admin staff postpone landing
landingStartTime		10:00	- calculation	- business variable Agreed landing time	Head of the technical department	it is reviewed annually

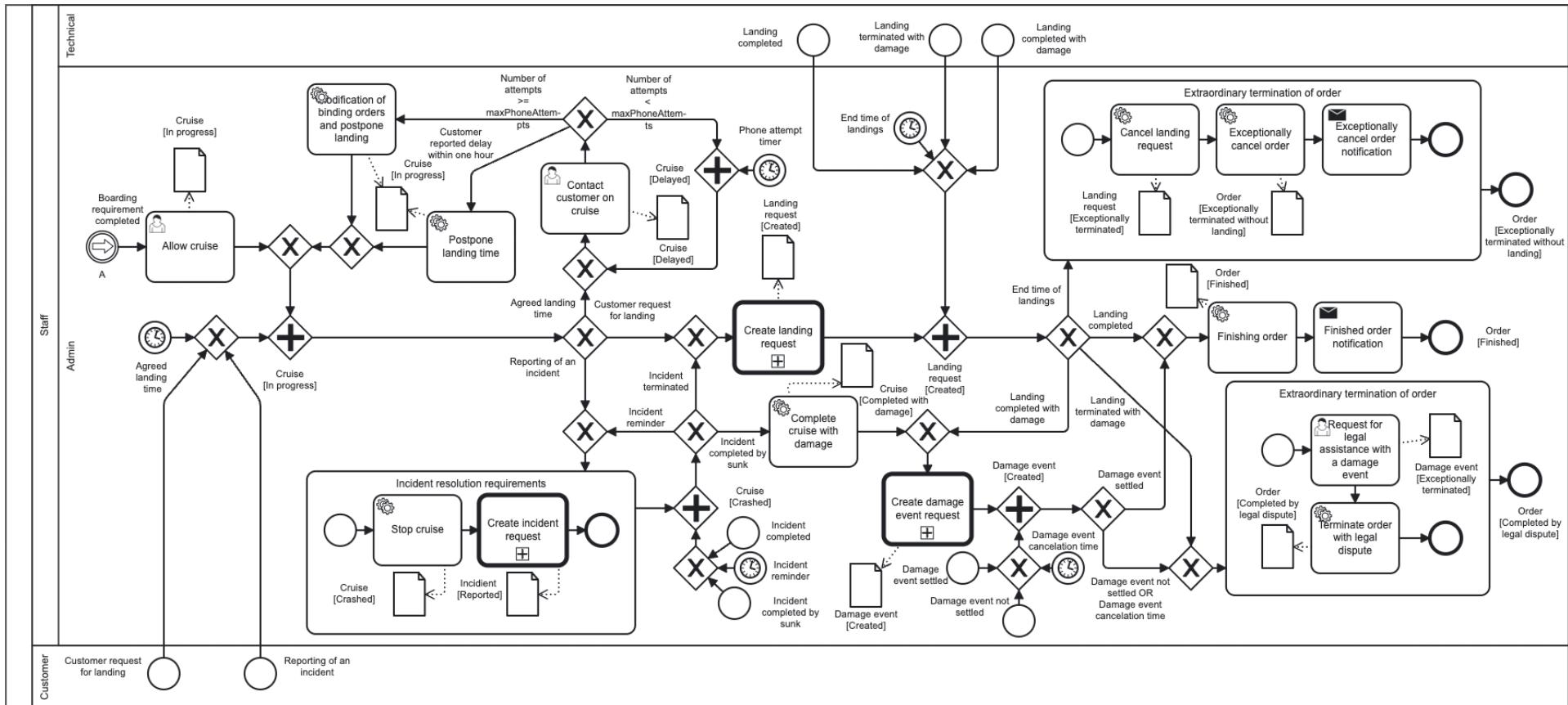
Phone attempt timer	1 hour	- time event	- key process state Cruise[Delayed]	Head of the technical department	has not changed for several years
maxPhoneAttempts	3 times	- task input	- Contact customer on cruise	Head of the technical department	has not changed for several years
End time of landings	time {landingEndTime} at date {LandingRequest.date}	- time event	- key process state LandingRequest[Created]	Head of the technical department and General business terms: Landing	
landingEndTime	16:00	- calculation	- business variable End time of landings	Head of the technical department	it is reviewed annually
Incident reminder	30 minutes and Incident state still [Reported]	- time event	- key process state Cruise[Crashed]	Internal policy for incidents	
<i>Name</i>	<i>Value</i>	<i>Usage</i>	<i>Location</i>	<i>Source</i>	<i>Notes</i>
Damage event cancellation time	90 days	- time event	- key process state DamageEvent[Created]	Internal policy for damage events	
seasonStartDate	1st of May at current year	- task input	Reconsider canceling	Internal business policy	
seasonEndDate	31st of September at current year	- task input	Reconsider canceling	Internal business policy	
maxBoardPostponeDays	3 days	- task input	Change boarding date	General business terms: Boarding	has not changed for several years
boardRefundPercentage	20%	- task input - notification	- Cancel order with penalty payment - task Cancel order with penalty payment notification	General business terms: Compensations	has not changed for several years

landPostponeDays	1 day	- task input	Modification of binding orders and postpone landing	General business terms: Landing	has not changed for several years
landPostponeTime	1 hour	- task input	Postpone landing time	Head of the technical department	could change after season
End time of boarding	{boardingEndTime} at current day	- time event - time event	- boarding process state Boarding request[Suspended] - boarding process state Boarding request[On inspection]	Head of the technical department and General business terms: Boarding	
Extended boarding cancellation	15 min	- time event	- boarding process state Boarding request[Extended]	General business terms: Boarding	it will not change
infoEmail	info@charter.com	- notifications	all	Head of IT department	
infoPhoneNumber	+420 999 999	- notification	from task Cancel order without refund notification	Head of administrative department	
generalConditionURL	charter.com/general-business-terms	- notification - notification	- Cancel order with penalty payment notification - Exceptionally cancel order notification	Head of IT department	
landRefundPercentage	25%	- notification	Exceptionally cancel order notification	General business terms: Compensations	has not changed for several years

Tabulka 4.17: List of business variables



Obr. 4.24: Technological model: process diagram of key process with business variable part 1



Obr. 4.25: Technological model: process diagram of key process with business variable part 2

#### 4.4.2 Špecifikácia notifikácií

Task	Receiver	Sender	Subject
Request deposit payment	{Customer.email}	{B:infoEmail}	Deposit required

**Text:**

We remind you that  
the deposit for the order n.{Order.number}  
must be paid by {B:Deposit reminder limit}.

The Charter company

Tabuľka 4.18: Notification for deposit payment

Task	Receiver	Sender	Subject
Request full price payment	{Customer.email}	{B:infoEmail}	Full price payment required

**Text:**

We remind you that  
full price ({Order.price}) for the order n.{Order.number}  
must be paid by {B:Deposit cancelation limit}.

The Charter company

Tabuľka 4.19: Notification for full price payment

Task	Receiver	Sender	Subject
Cancel reservation notification	{Customer.email}	{B:infoEmail}	Cancel reservation

**Text:**

Unfortunately, I have to inform you  
that we canceled your order n.{Order.number}  
because you did not pay the deposit.

The Charter company

Tabuľka 4.20: Notification for cancel reservation

Task	Receiver	Sender	Subject
Cancel order without refund notification	{Customer.email}	{B:infoEmail}	Cancel order

**Text:**

Unfortunately, I have to inform you  
that we canceled your order n.{Order.number} without refundation.  
For further information, contact  
the customer department at {B:infoPhoneNumber}.

The Charter company

Tabulka 4.21: Notification for cancel reservation without refund

Task	Receiver	Sender	Subject
Completed order notification	{Customer.email}	{B:infoEmail}	Completed order

**Text:**

Your order n.{Order.number} has been paid and completed.  
We look forward to your arrival on {Order.cruiseStartAt}  
Arrival for boarding is required from {B:boardingStartTime} to {B:boardingEndTime}.

The Charter company

Tabulka 4.22: Notification for completed order

Task	Receiver	Sender	Subject
Change boarding date notification	{Customer.email}	{B:infoEmail}	Boarding date changed

**Text:**

Boarding date for order n.{Order.number}  
has been changed to {Order.movedBoardingDate}.  
Arrival for boarding is required from {B:boardingStartTime} to {B:boardingEndTime}.

The Charter company

Tabulka 4.23: Notification for changed boarding date

Task	Receiver	Sender	Subject
Cancel order with penalty payment notification	{Customer.email}	{B:infoEmail}	Order exceptionally cancelled
<b>Text:</b>			
<p>We are sorry, but your order n.{Order.number} has been exceptionally cancelled.</p> <p>We apologize for the caused problems. You will be refunded full price plus {B:boardRefundPercentage} of the rental price and you are entitled to adequate financial compensation according to the {S:generalBusinessConditions}.</p>			
The Charter company			
Special Variable	Text	Type	Source
generalBusinessConditions	general business condition	link	{B:generalConditionURL}

Tabulka 4.24: Notification for cancel order with penalty

Task	Receiver	Sender	Subject
Exceptionally cancel order notification	{Customer.email}	{B:infoEmail}	Landing exceptionally unrealized
<b>Text:</b>			
<p>We apologize that we could not make the planned landing.</p> <p>You will be contacted by phone regarding further information.</p> <p>We apologize for the caused problems. You will be refunded full price plus {B:landRefundPercentage} of the rental price and you are entitled to adequate financial compensation according to the {S:generalBusinessConditions}.</p>			
The Charter company			
Special Variable	Text	Type	Source
generalBusinessConditions	general business condition	link	{B:generalConditionURL}

Tabulka 4.25: Notification for exceptionally canceled order

Task	Receiver	Sender	Subject
Finished order notification	{Customer.email}	{B:infoEmail}	Boarding date changed

**Text:**  
Your rental (order n.{Order.number}) has been successfully completed.  
We look forward to your next cruise on one of our ships.

The Charter company

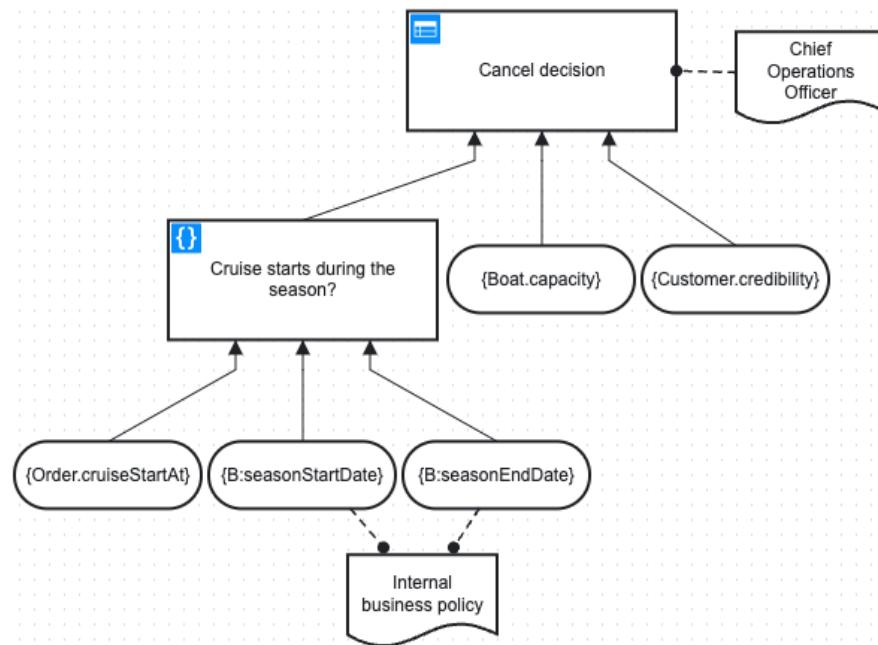
Tabuľka 4.26: Notification for finished order

Pri vytváraní notifikácií sme natrafili na výskyt špeciálnych premenných (tabuľky č.4.24 a č.4.25) v podobe odkazov na webové stránky. Zároveň sme identifikovali aj výskyt 4 nových business premenných, ktoré sme doplnili do zoznamu business premenných (na konci tabuľky č.4.17).

#### 4.4.3 Pridanie rozhodovacích modelov a tabuľiek

Jediné komplexné rozhodnutie v procese je rozhodnutie o zrušení objednávky v prípade, že nebola prijatá plná platba za objednávku mesiac pred plavbou v úlohe *Reconsider canceling*. V diagrame sa nenachádza žiaden doprevádzajúci text, preto by sme museli toto rozhodnutie konzultovať s vlastníkom procesu.

Po prediskutovaní business rozhodnutia pokračujeme vytváraním DRD diagramu. Identifikovali sme vstupy, ktoré tvoria atribúty troch objektov a dva business premenné.



Obr. 4.26: Technological model: DRD for Cancel decision

S vlastníkom procesu, ktorý je zároveň prevádzkový riaditeľ, sme si vyjasnili logiku rozhodnutia. Logiku rozhodnutia sme zaznamenali v podobe rozhodovacej tabuľky.

Cancel decision   Hit Policy: First					Annotations
	When Cruise starts during the season? boolean	And {Boat.capacity} number	And {Customer.credibility} number	Then Cancel decision boolean	
1	true	]3..6]	>= 85	false	Good credibility even for popular boats during season.
2	true	]3..6]	< 85	true	Not enough credibility even for popular big boats during season.
3	true	-	>= 50	false	Enough credibility for less charterable boats in season.
4	true	-	< 50	true	
5	false	<= 3	>= 50	true	Small boats are popular off-season.
6	false	> 3	-	false	Big boats are not so popular off-season.

Obr. 4.27: Technological model: decision table

Rozhodnutie ovplyvňuje dátum začiatku plavby, kapacita lode a kredibilita zákazníka.

Mimo sezónu pre lode, ktoré nezvyknú byť vyťažené nie je potrebne objednávku rušiť. Naopak aj v sezóne pri objednávke na obľúbene lode sa môžeme na zákazníka spôsobovať ak už má s podnikom nejakú história a podnik ho považuje za kredibilného. Tabuľka nepokrýva všetky intervaly, preto je zvolená stratégia výberu prvého (*Policy: First*). Ak teda hodnoty zapadajú do viacerých množín, vyberie sa prvé pravidlo.

# **5. Transformácia technologického modelu na implementačný model**

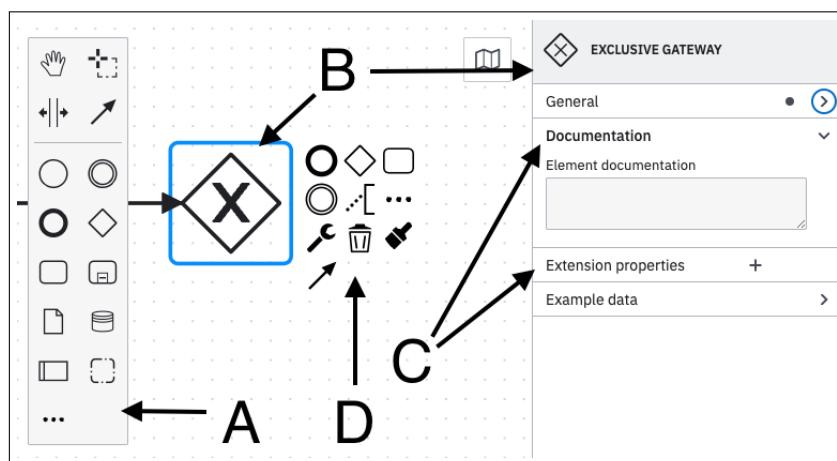
V tejto kapitole navrhнемe metodický postup transformácie technologického modelu do implementačného modelu pre platformu Camunda. V druhej časti kapitoly tento metodický postup aplikujeme na technologický model z kapitoly č.4. V poslednej časti kapitoly v krátkosti zhodnotíme používateľské dojmy z nástroja Camunda.

## 5.1 Metodický postup transformácie do implementačného modelu nástroja Camunda

Metodický postup sa skladá z krokov, ktoré sa vykonávajú len raz a krokov, ktoré sa opakujú na menšie pracovné celky. Kroky aplikovateľné na menšie pracovné celky sa obvykle skladajú z dvoch častí. Modelovania a konfigurácie. Nie je nevyhnutné ich vykonávať bezprostredne za sebou. Preto sú modelovacie kroky označené číslami (1., 2., atď.) a konfiguračné kroky písmenami (A., B., atď.).

Doporučujeme najprv postupovať v poradí akom je metodicky postup napísaný, teda po modelovaní pokračuje konfiguráciou (1., A., 2., B., atď.). Neskôr môžeme zvážiť či nám takýto spôsob práce vyhovuje. Nie je problém vykonať najprv modelovacie kroky (1., 2., atď.). Nástroj nám obvykle pri nenakonfigurovaných elementoch bude zobrazovať chybové správy, ktoré nám budú pripomínať aby sme po modelovaní nezabudli na konfiguráciu.

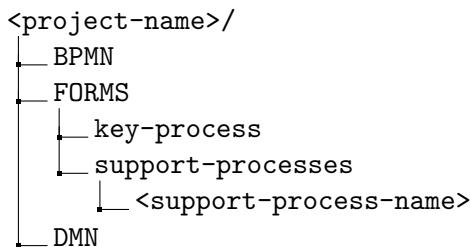
V popise vykonávaných krokov sa objavujú veľké písmena, ktoré odkazujúce na písmená v obrázkoch doplňujúcich text. Obrázky sa nachádzajú pod textom alebo medzi odstavcami textov. Elementy sa vkladajú pomocou bočnej ponuky (A). Po kliknutí na element sa zobrazujú jeho nastavenia (B). Nastavia sa skladajú zo záložiek (C) a ponuky elementu (D).



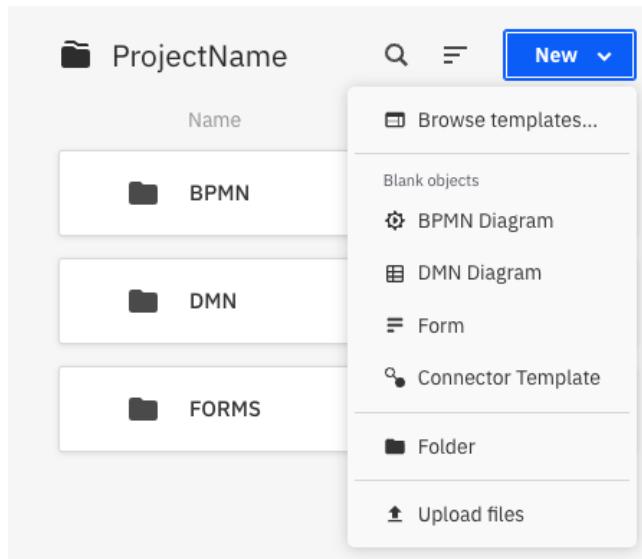
Obr. 5.1: Vysvetlivky

### 5.1.1 Vytvorenie štruktúry projektu

Vytvoríme vhodnú projektovú štruktúru pomocou priečinkov. Pri Self-managed verzii to musíme urobiť priamo v prostredí operačného systému. Navrhovaná štruktúra priečinkov je nasledovná.



Do priečinkov podľa významu názvu budeme vkladať jednotlivé súbory diagramov a formulárov. SaaS verzii ponúka na túto správu projektových súborov príjemné užívateľské rozhranie v online modeléry (obrázok č.5.2).

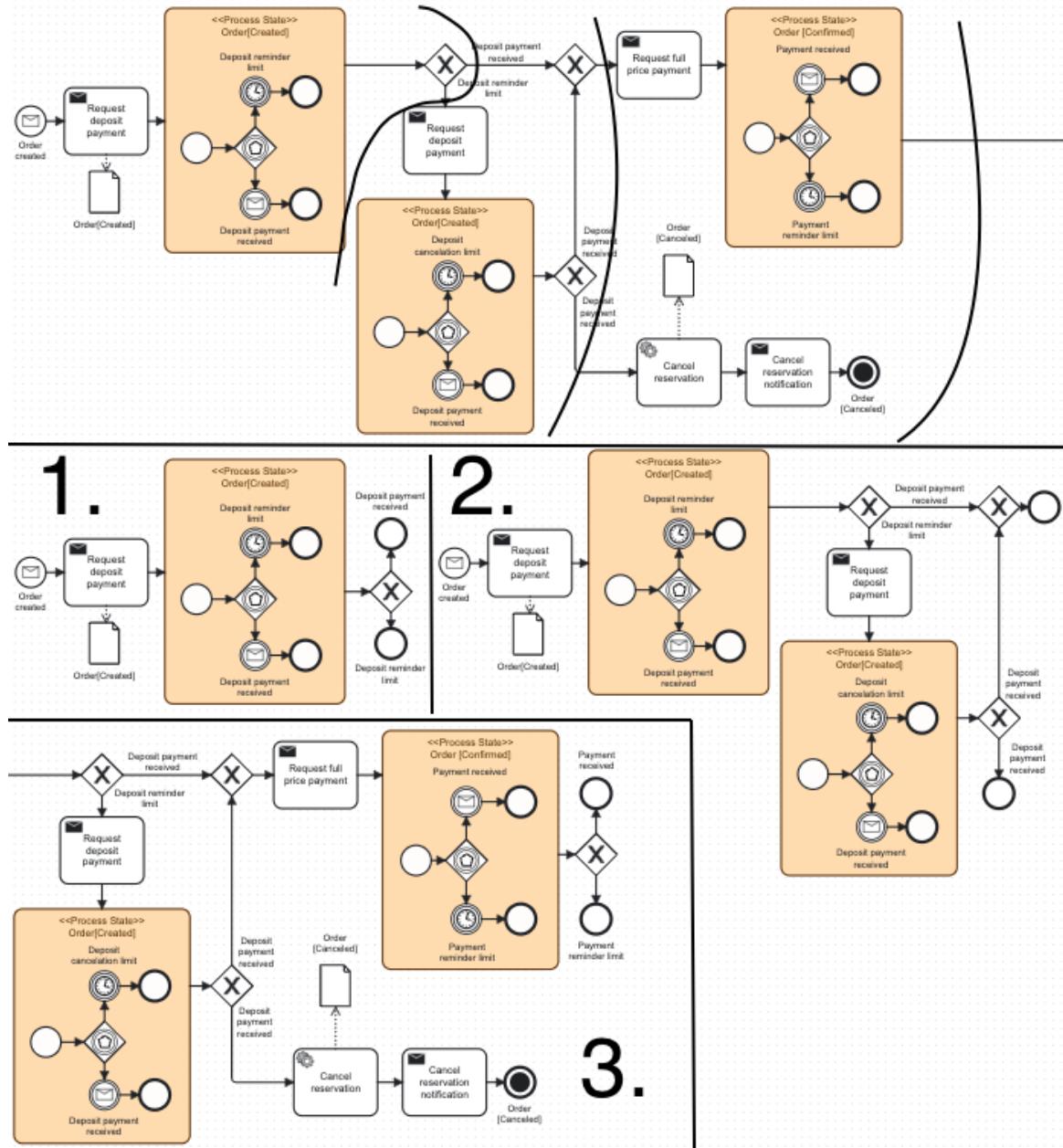


Obr. 5.2: Online Modeler SaaS verzie

### 5.1.2 Výber pracovnej jednotky

Tak ako pri konceptuálnom modelovaní je aj pri implementačnom modelovaní potreba dokomponovať. Pred začatím je potrebné na základe veľkosti a komplikovanosti procesu premyslieť na aké menšie celky proces rozdelíme. Následné budeme aplikovať ďalšie kroky na tieto menšie celky, ktoré sme pomenovali ako **pracovné jednotky**. Tie následne nasadíme a otestujeme ich funkčnosť. Potom môžeme rovnakým spôsobom pokračovať na ďalšie časti procesu. Tento prístup k vytvárané implementačného modelu nie je povinný ale veľmi výrazne doporučený. Výrazne zjednoduší celý vývoj a hľadanie chýb.

Doporučená veľkosť pracovnej jednotky je procesný stav s úlohami, ktoré proces do tohto stavu dostanú a brána, ktorá za nim nasleduje. Po získaní skúsenosti s nástrojom Camunda je možné pracovnú jednotku primeraným spôsobom zväčšiť. Určite ale nedoporučujeme modelovať naraz celý proces, ak sa nejedná a veľmi jednoduchý proces. Ukážka rozdelenie procesu na menšie postupne implementované pracovné jednotky vidíme na obrázku č.5.3.



Obr. 5.3: Dokompozícia implementácia

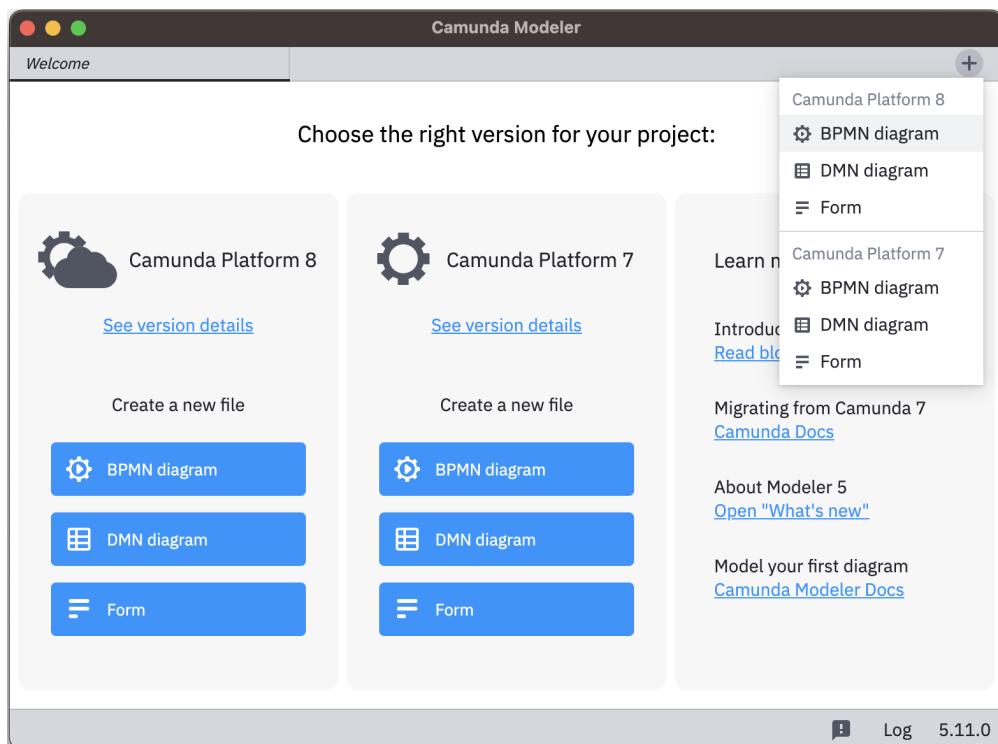


### 5.1.3 Vytvorenie procesného diagramu

Tento krok opakujeme pre všetky procesy v technologickom modeli.

#### 1. vytvorenie BPMN diagramu

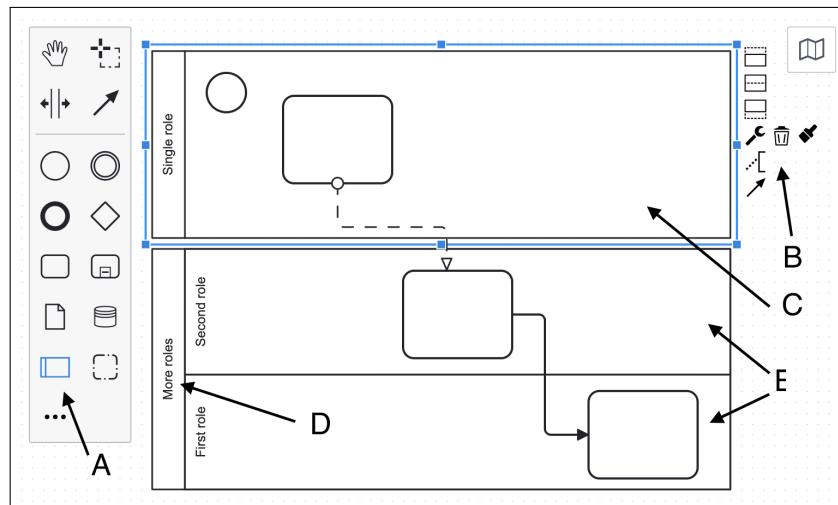
Vytvoríme súbor BPMN diagram pre verziu 8. Ten uložíme s názvom procesu do priečinku na doporučenú cestu (<project-name>/BPMN/<diagram-name>.bpnn) v projektovej štruktúre.



Obr. 5.4: Vytvorenie BPMN diagramu

## 2. pridanie plavebných dráh

V nástroji Camunda majú plavebné dráhy viac vizuálny charakter ako funkčný. Zabraňujú sice priamej komunikácií medzi dráhami ale nedokážu funkčne vymedziť úlohy v dráhach pre technické role.



Obr. 5.5: Pridanie plavebných dráh

Dráhu pridáme pomocou tlačidla A. Pomocou tlačidiel B potom môžeme upravujeme množstvo vnútorných dráh. Pri vytváraní viacerých dráh (pomocou tlačidla A) sa dátia medzi dráhami (C a D) priamo nevymieňajú. Je potom potrebné implementovať logiku vymieňania správ.

Vhodnejším spôsobom je vytvorenie jednej dráhy (A) a následné rozdelenie (B) na ďalšie vnútorné dráhy (E). V takom prípade sa neporušujú flow procesu a úlohy si medzi sebou vymieňajú dátá.

Vzhľadom nato, že sa element procesného stavu bude veľmi podobný ako stav v konceptuálnom modeli, **nevytvárame plavebné dráhy pre technické role, ktoré majú iba udalosti a nemajú úlohy**.

### 3. konfigurácia vstupnej udalosti

Vstupná udalosť sa v diagrame procesu nachádza už po jeho vytvorení. Ak sa proces spúšta manuálne, ponecháme obyčajnú štartovaciu udalosť. Pri spustení procesu v dôsledku nejakej udalosti nastavíme štartovaciu udalosť na typ *Message Start Event* a ak sa spúšta časom tak nastavíme typ na *Timer Start Event*.

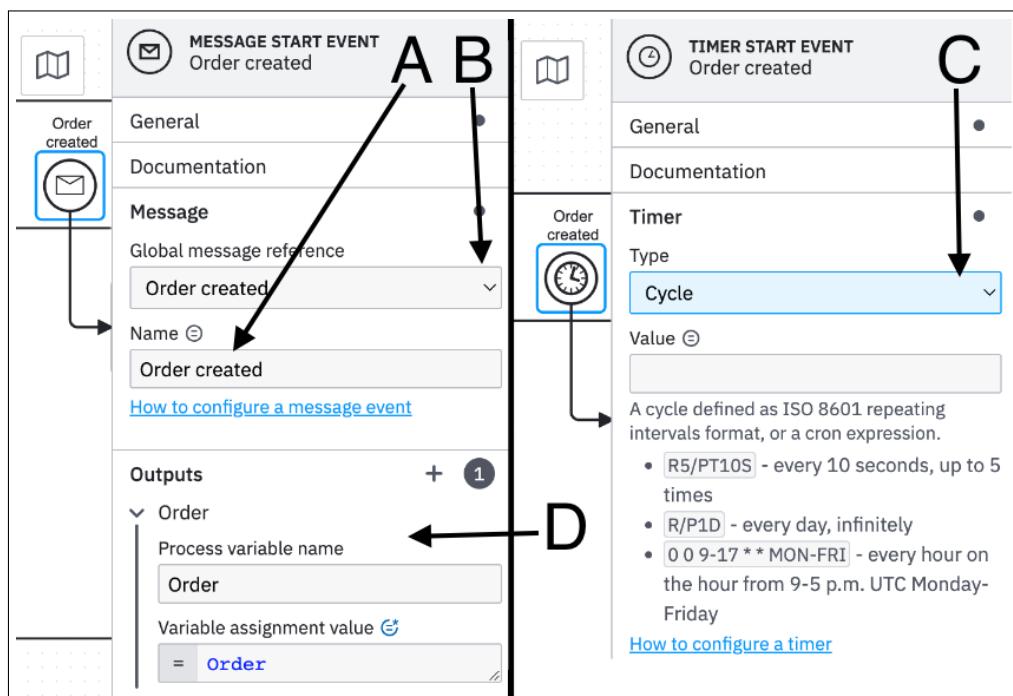
#### *Message Start Event*

Pri tomto type je potrebné vytvoriť názov správy spúšťacej udalosti (A), ktorý slúži ako identifikátor danej udalosti. Ak daná udalosť už bola vytvorená skôr, nájdeme ju v zozname správ (B). Ďalej je veľmi vhodné namapovať očakávané vstupy z udalosti na premenné v procese (D). Má to dve výhody. Prvou je prehľad o tom, aké premenné sa v procese nachádzajú. Druhou je vznik tzv. whitelist/allowlist, teda zoznamu povolených premenných vstupujúcich do procesu. To znamená, že ak vytvoríme takéto mapovanie, žiadna iná premenná nebude v procese dostupná.

#### *Timer Start Event*

Pri tomto type je potrebné zvoliť spôsob časovača. Prvou možnosťou je cyklické spúšťanie udalosti. Druhou možnosťou je konkrétny dátum spustenia. Pri oboch variantách je potom nutné vyplniť hodnotu časovača ([Timer events 2023](#)) vo formáte zhodnom s ISO 8601. Pre dátum je to napr. *2019-10-02T08:09:40+02:00*, a pre cyklus je to napr.

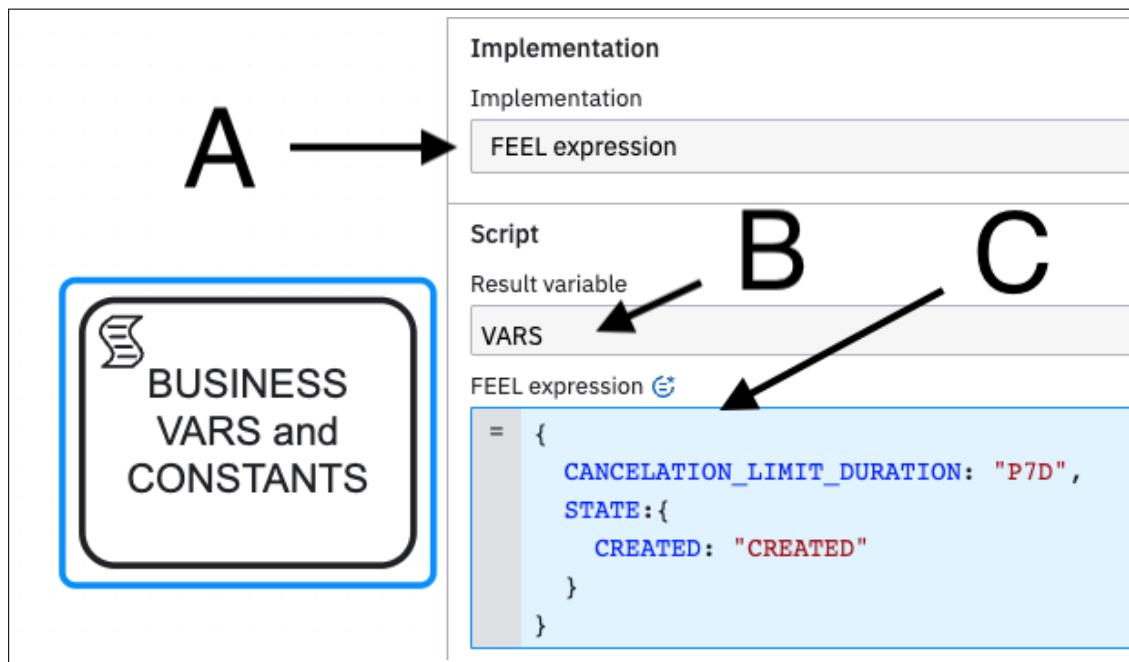
*R3/2022-04-27T17:20:00Z/P1D*.



Obr. 5.6: Modelovanie spúšťacích a ukončovacích udalostí

#### 4. pridanie zoznamu premenných a konštánt

Po štartovacej udalosti, ktorá je v diagrame automaticky pri jeho vytvorení, vytvoríme úlohu typu *Script Task*. V nastavenia zvolíme typ implementácie *FEEL expression* (A). Nastavíme vhodný názov, pod ktorým budeme v procese pristupovať k premenným a konštantám (B). Do tela skriptu vložíme JSON objekt, ktorého atribúty budú predstavovať business premenné alebo iné dôležité premenné či konštanty, ktoré je dobré mať na jednom mieste (napríklad hodnoty stavov).



Obr. 5.7: Zoznam premenných a konštánt

Tento zoznam je vhodné napĺňať postupe pri modelovaní a konfigurovaní procesu. V procese potom nepoužívame statické hodnoty ale hodnoty vkladáme pomocou vytvoreného objektu (`VARS.<nazov_premennej>`).

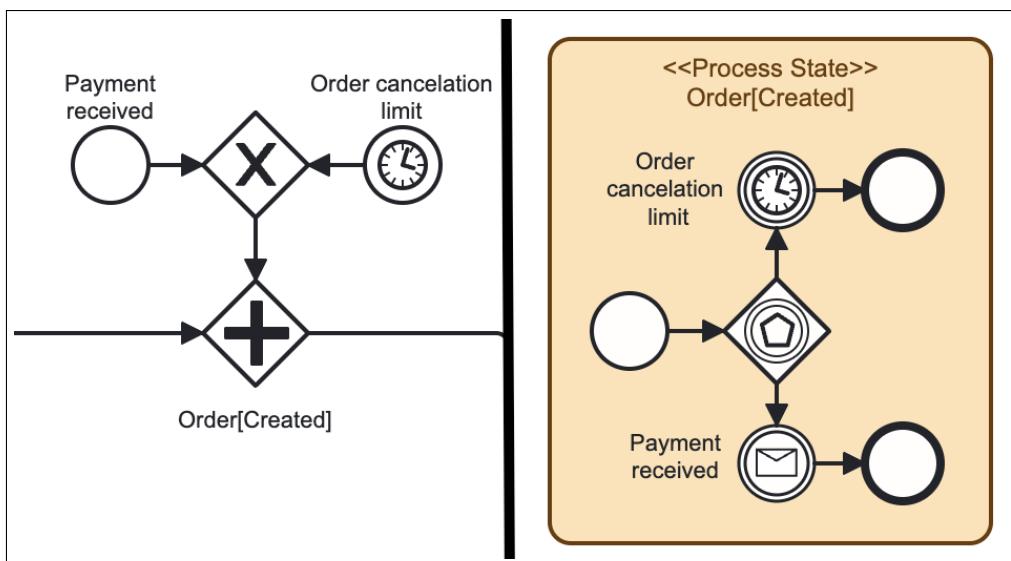
Úlohe môžeme pridať premenné alebo konštanty aj v podobe výstupov z úlohy, v záložke *Outputs*. Ak ale už používame výstup skriptu. Musíme ho taktiež uviesť medzi výstupné premenné. Zavedie výstupných premenných totiž obmedzuje všetky ostatné výstupy úlohy.

#### 5.1.4 Transformácia procesu

Tento krok vykonávame na pracovnej jednotke. Ak by pri transformácii došlo k nejakým problémom, ktoré nemajú jasné a jednoznačné riešenie doporučujeme sa pozrieť do sekcie 5.1.9, kde popisujeme isté špecifické problémy, ktoré sme objavili a doporučenia na ich riešenie.

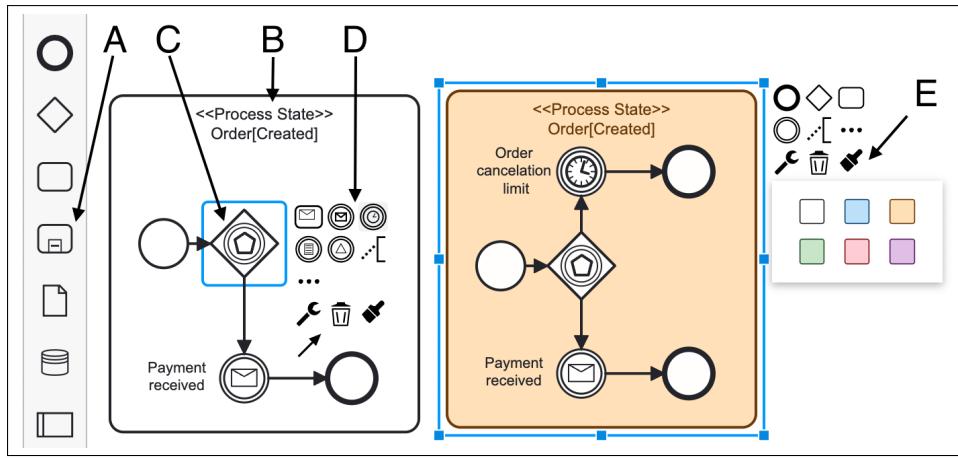
##### 1. modelovanie procesného stavu

Prvým elementom, ktorý budeme do implementačného diagramu pridávať (okrem plavebných dráh) je procesný stav. Okrem toho, že procesné stavy sú významovo veľmi dôležité, budú predstavovať najväčšie stavebné elementy procesného diagramu. Spôsob transformácie procesného stavu už definoval Řepa (2022, str. 28). Transformáciu vidíme na obrázku č. 5.8.



Obr. 5.8: Ukážka transformovaného stavu

Postup vidíme na obrázku č.5.9 a jeho popis je nasledovný. Procesný stav bude zabalený do elementu *expanded Sub-process* (A), ktorému pridáme stereotyp «*Process State*» a názov procesného stavu (B). Vo vnútri používame *Event-based gateway* (C), ktorá čaká na udalosti typu *timer intermediate catch* pre časové udalosti a *message intermediate catch* pre ostatné udalosti, ktoré pridávame pomocou bočnej lišty modeléru alebo ponuky brány (D). Ďalším krokom je farebné odlišenie stavového elementu (E).



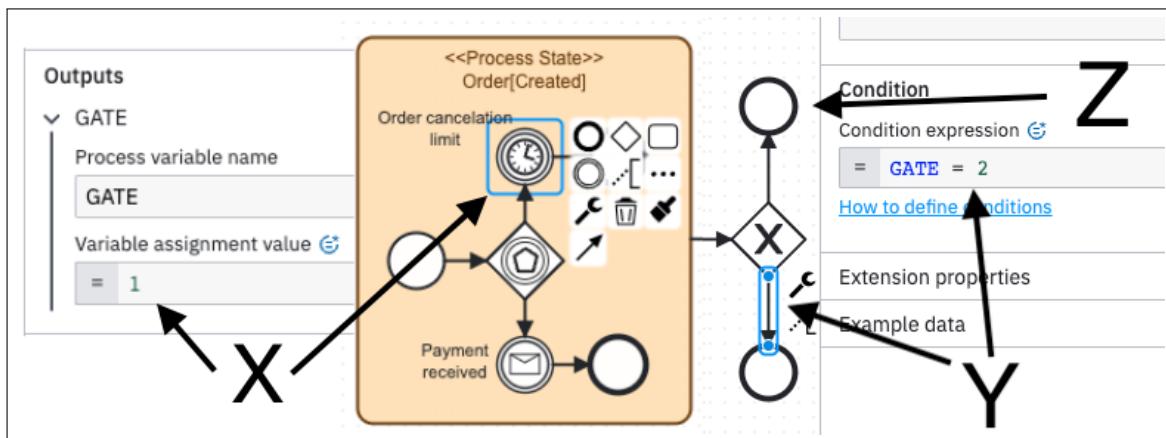
Obr. 5.9: Transformácia stavov procesu

Za element procesného stavu pridáme *Exclusive gateway* a z nej vytvoríme rovnaký počet cest ako je počet udalostí. Cesty zakončíme *End Event* elementom (Z na obrázku č.5.10).

#### A. konfigurácia výstupu zo stavu a stavových udalostí

##### Cesty z Exclusive gateway

V cestách vychádzajúcich z brány musíme nastaviť podmienok. Aby sme mohli odlišiť, ktorá udalosť nastala v elemente procesného stavu, musíme udalostiam nastaviť pomocnú výstupnú premennú (X) voči, ktorej definujeme podmienky na cestách vychádzajúcich z brány (Y).



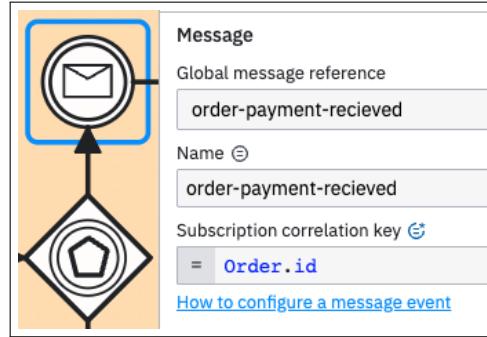
Obr. 5.10: Rozlíšenie udalostí mimo elementu procesného stavu

##### Timer Intermediate Catch Event

V tejto udalosti musíme nastaviť v záložke *Timer* hodnotu parametru *Duration*. Hodnota musí byť vo formáte pre dobu trvania zhodnom ([Timer events 2023](#)) s ISO 8601 (napr. *PT15S* alebo *P14D*).

### *Message Intermediate Catch Event*

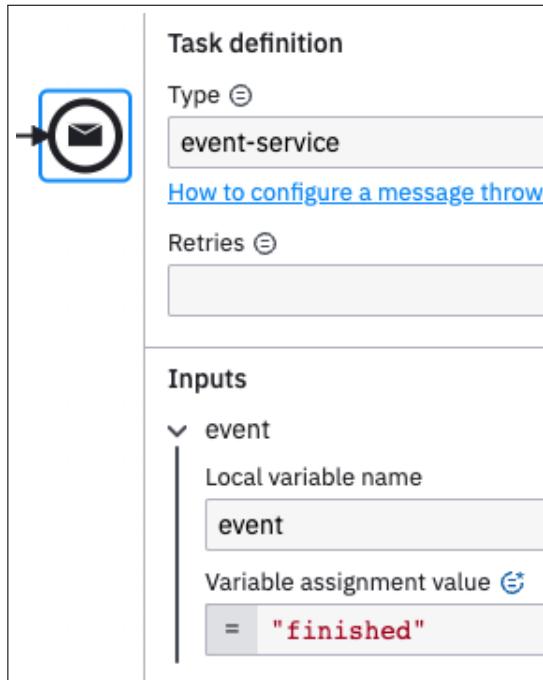
V tejto udalosti musíme nastaviť v záložke *Message* hodnotu parametru *Name*, ktorá predstavuje identifikátor správy. Ďalšou povinnou hodnotou je *Subscription correlation key*. Ide o identifikátor priradenie správy ku konkrétnnej inštancii procesu. Vhodným korelačným klúčom sú identifikátory hlavných objektov, ktoré sa v procese používajú. Názov správy a korelačný klúč sa budú používať v servisoch, ktoré správu posielajú.



Obr. 5.11: Message Intermediate Catch Event

### *Koncová udalosť*

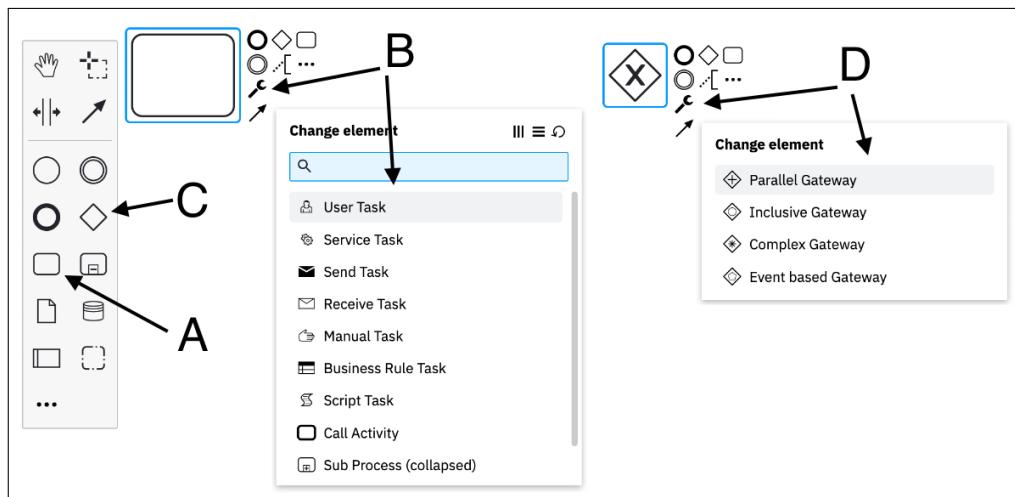
Za normálnych okolností nie je potrebná žiadna konfigurácia. Môže sa ale stať, že koncová udalosť je v podpornom procese a je potrebné na konic vytvoriť udalosť. V takom prípade použijeme *Message End Event* a v nastaveniach udalosti definujeme, ktorá služba bude za zaslanie udalosti zodpovedná.



Obr. 5.12: Message End Event

## 2. modelovanie úloh a brán

Vytvoríme všetky úlohy z technologického modelu pomocou tlačidla A. Vyberieme príslušný typ danej úlohy cez nastavenia úlohy (B). Vytvoríme všetky brány z technologického modelu pomocou tlačidla C a vyberieme im príslušné typy cez nastavenia brány (D).



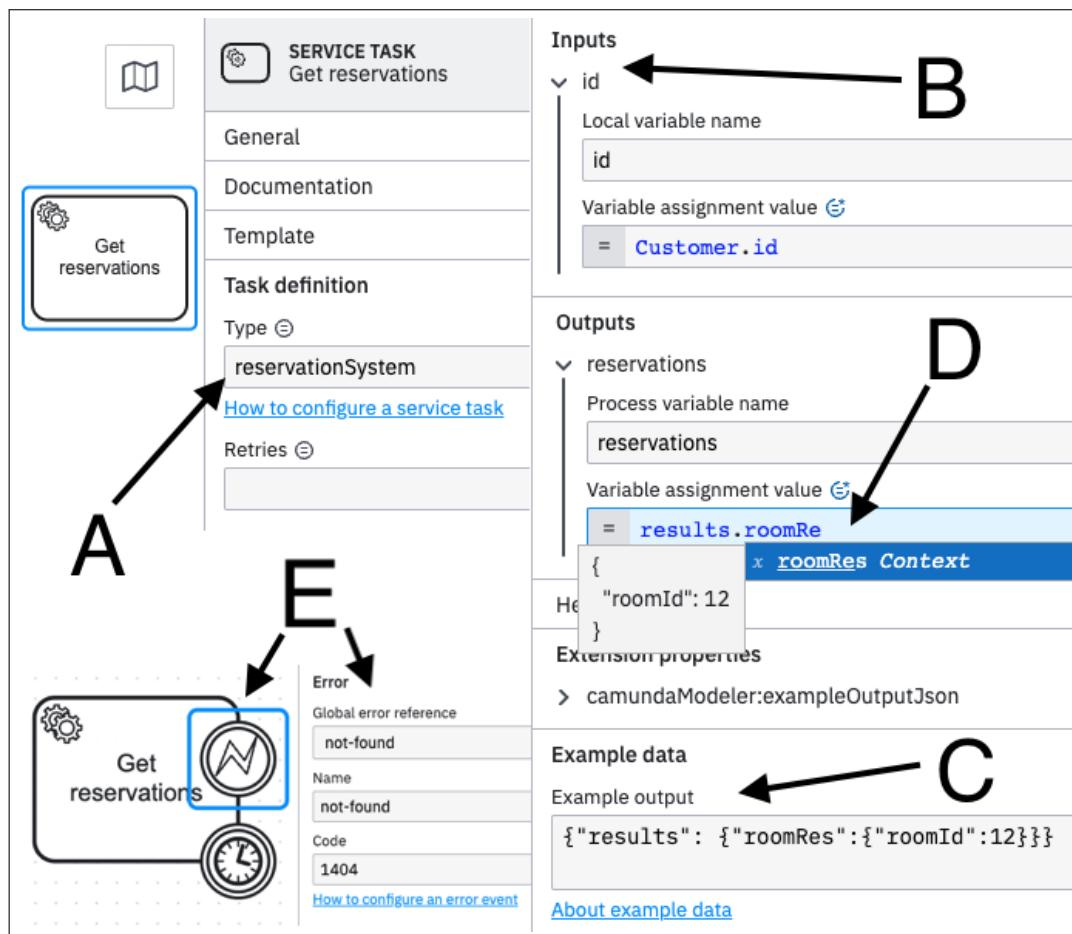
Obr. 5.13: Modelovanie úloh a brán

Následne spojíme všetky úlohy, brány a procesný stav šípkami podľa technologického modelu.

## B. konfigurácia úloh a brán

### Service Task

Úloha slúži na volanie služby, ktorá je implementovaná mimo nástroja, a ktorá s Camundou komunikuje pomocou Zeebe klienta. Je nutné definovať názov služby (A), ktorý bude následne používaný ako identifikátor služby v zdrojovom kóde. Pri potrebe poslat nejaké vstupné parametre službe vyplníme vstupy v nastaveniach (B). Pokiaľ nám služba vracia nejaké hodnoty môžeme do nastavení vložiť ukážkový formát priatých dát (C) a následne využívať našepíváč (D) pri definovaní výstupov z úlohy.



Obr. 5.14: Service Task

Pri volaní služieb môže dochádzať k rôznym chybám. Môže napríklad zlyhať validácia alebo službe posielame chybné dátá. Môže ale dôjsť aj k prerušeniu komunikáciu, čo v sietovej komunikácii nie je výnimočná vec. Je preto potrebné zistiť aké predpokladané chyby môžu v službe nastaviť, o ktorých služba dokáže informovať v podobe vrátenia chybového kódu. Tieto informácie by sa mali nachádzať v špecifikácii rozhrania služby. Všetky takéto chyby je potom vhodne odchytávať pomocou *Error Boundary Event* elementu (E), ktorému nastavíme názov a kód chyby.

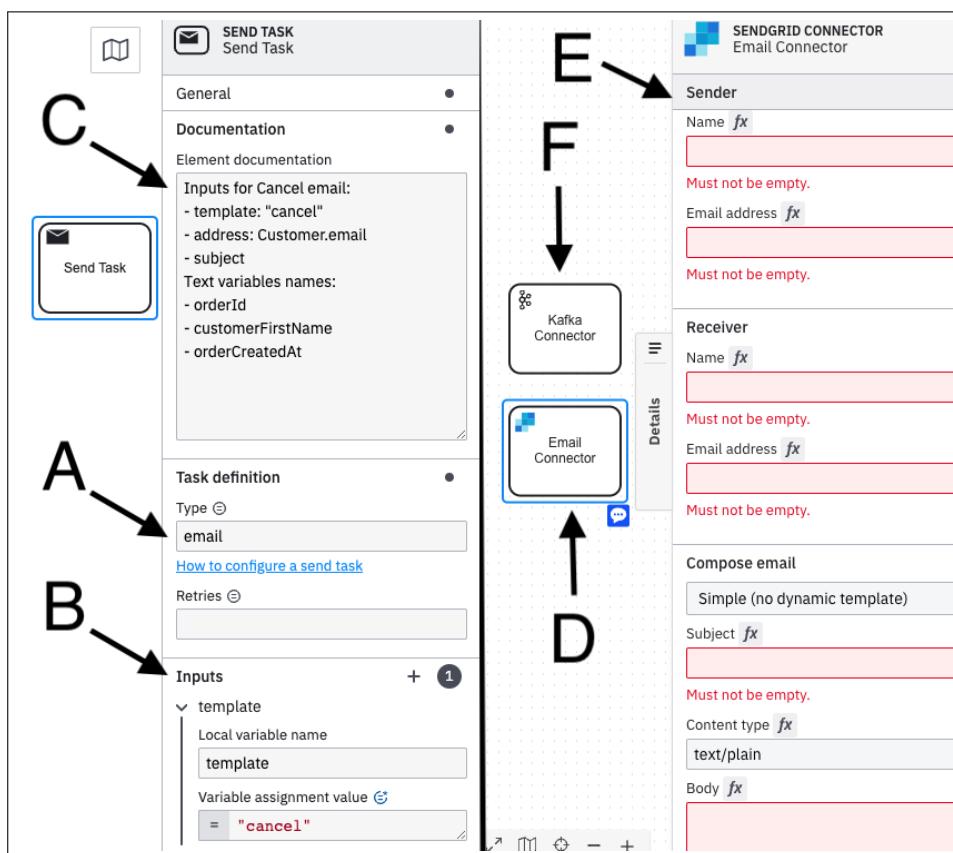
Ďalej môže dôjsť ku chybe, ktorá nie je reprodukovaná zo služby naspäť do procesu. Úloha tak nie je ukončená a proces stojí. Preto je vhodné zvážiť použitie časovača v podobe *Timer boundary* elementu (E), ktorým tento scenár vyriešime.

Posledným dôležitým problémom na, ktorý nesmieme zabudnúť je zlyhanie pri pokuse vytvoriť so službou spojenie. Tu nám nastavenia úlohy ponúkajú parameter *Retries*, ktorý specifikuje počet pokusov zavolania služby, ak sa so službou nepodarilo spojiť. Camunda už ale má predvolenú hodnotu (aj keď nie je priamo uvedená) v podobe troch pokusov.

## Send Task

Táto úloha je prakticky iba graficky odlišený *Service Task*. Je teda potrebné definovať názov služby (A) a funkcia odosielania musí byť implementovaná pomocou tejto služby. Obyčajné textové parametre (adresa alebo predmet e-mailu) je možné vložiť pomocou vstupov do úlohy (B). Komplikovanejšie vstupy (napr. texty e-mailov) je vhodnejšie definovať v službe a dohodnúť spôsob predávania dát. Tento spôsob si môžeme poznačiť do záložky pre dokumentáciu (C).

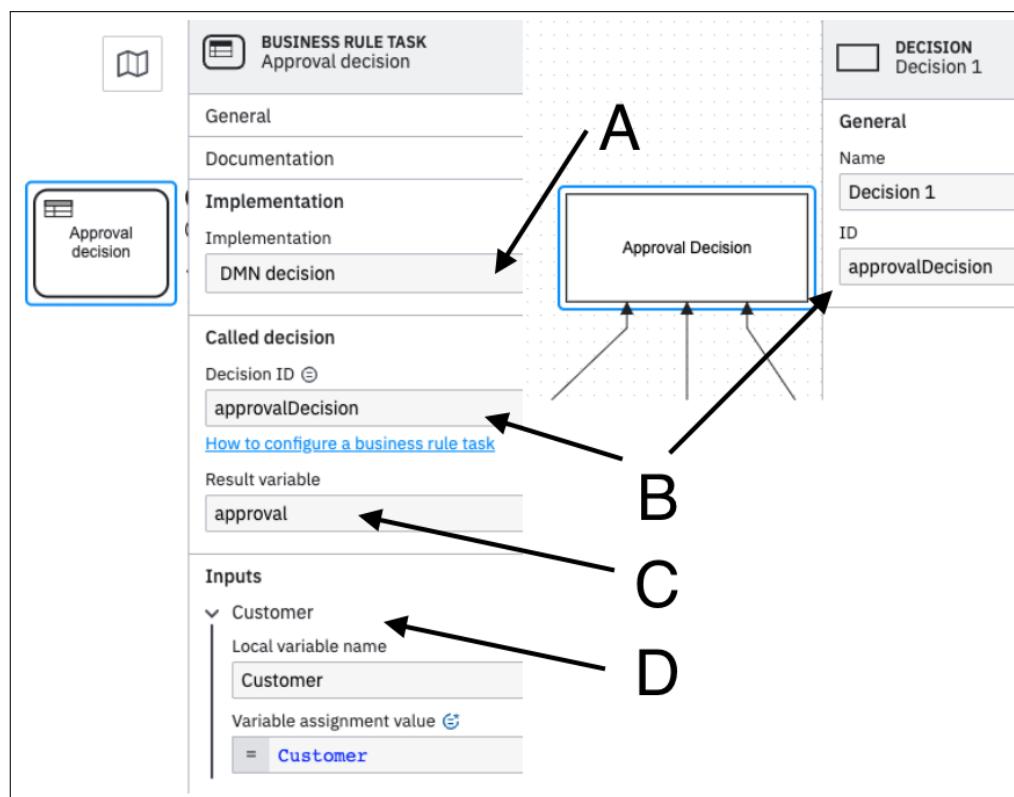
V SaaS verzii existuje *SendGrid Connector*, ktorý slúži ako prepojenie na externú e-mailovú službu (D). Po registrácii v službe je možné vygenerovať API kľúč a potom môžeme konfigurovať e-mail priamo v modeléri (E). Ďalším konektorom, ktorý ponúka SaaS verzia je *Kafka Connector* (F).



Obr. 5.15: Send Task

### Business Rule Task

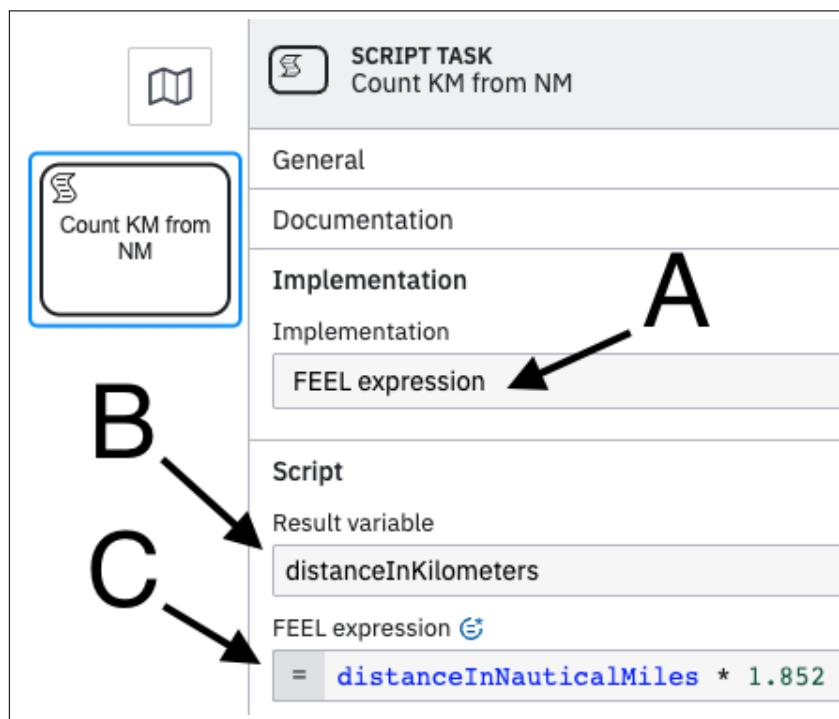
Úloha sa používa pri vykonávaní business rozhodnutí podľa stanovených pravidiel. Vzhľadom na podporu DMN v nástroji Camunda je primárnym použitím implementácia pomocou *DMN decision* (A). Tu musíme definovať Decision ID odpovedajúci identifikátoru rozhodnutia v DRD diagrame (B), ktorého tvorbe sa venujeme v sekcií 5.1.6. Výsledok rozhodnutia sa vloží do premennej ktorej názov musíme definovať (C). Do business rozhodnutí obvykle vstupuje viacero parametrov. V prípade potreby definujeme, ktoré dátá z procesu budú vstupmi do rozhodnutia (D). Ak na rozhodnutie nepoužívame DMN, môžeme si vybrať variantu implementácie *Job Worker*. V takom prípade funguje úloha rovnako ako *Service Task* a ide iba o vizuálne odlišenie od ostatných úloh.



Obr. 5.16: Business Rule Task

### Script Task

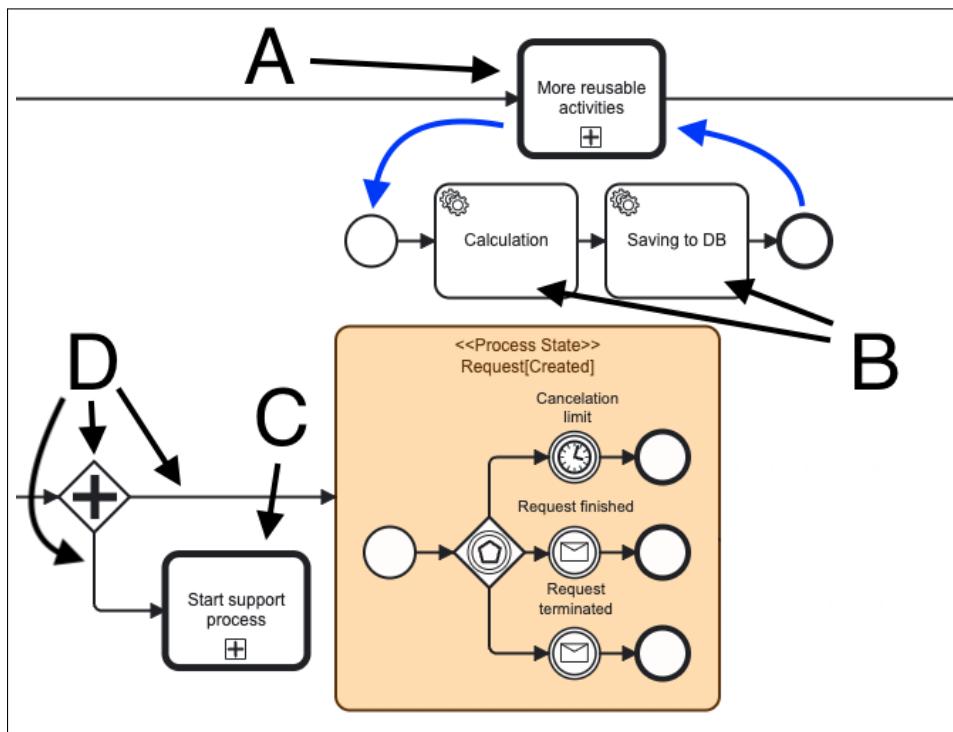
Úloha môže byť používa dvoma spôsobmi, ktoré si vyberáme v nastaveniach (A). Z princípu úlohy je primárnym použitím vykonanie jednoduchého skriptu. Ten je možné napísat pomocou jazyka FEEL([Decision Model and Notation 2023](#), str. 79–162). Skript vytvorí premennú, ktorá bude výstupom z úlohy. Je teda potrebne definovať názov premennej (B) a FEEL výraz (C). Druhým použitím úlohy je *Job worker*. V takomto prípade sa úloha správa rovnako ako *Service Task*. Význam takého použitia môže byť grafické odlišenie rôznych typov volaní externých služieb (napr. interná/externá služba alebo služba pracujúca s databázou/bez databázy).



Obr. 5.17: Script Task

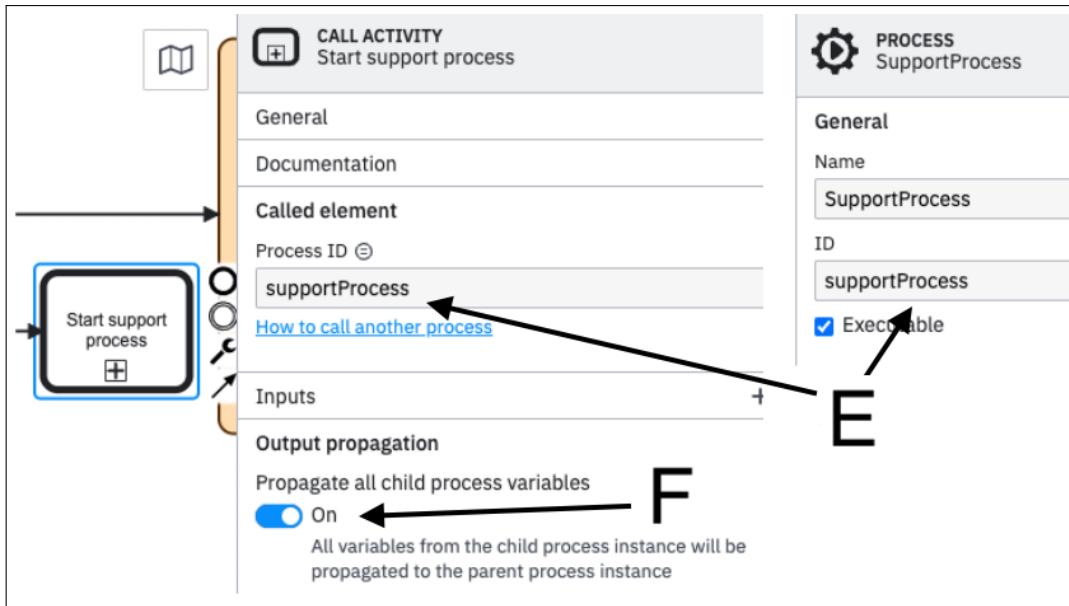
### Call Activity

Táto úloha sa používa na znova použitie (A) niekoľko menších úloh (B) vykonávajúcich nejakú spoločnú funkcia. Druhým spôsobom použitia je volanie podporného procesu (C) s následným čakaním na jeho výsledok. Ak ide o spustenie podporného procesu musíme pred takéto volanie doplniť AND bránu, z ktorej ide šípka do volania a do procesného stavu (D). Ak by sme v diagrame ponechali šípku z volania a nevytvorili AND bránu, procesnom stave by nebolo možné odlišiť ako podporný proces skončil, pretože by sa workflow dostal do procesného stavu až po skončení úlohy.



Obr. 5.18: Call Activity 1

V oboch prípadoch použitia musíme vyplniť parameter *Process ID* v záložke *Called element* v nastaveniach úlohy. Ide o identifikátor, ktorý musí byť zhodný s identifikátorom *ID* v záložke *General* v podpornom procese (E). Spustenie podporného procesu môžeme realizovať aj bez použitia *Call Activity* pomocou udalostí. Veľkou výhodou tohto elementu je ale možnosť propagácia výstupov z podporného procesu do hlavného procesu, ktorú povolíme v nastaveniach (F).

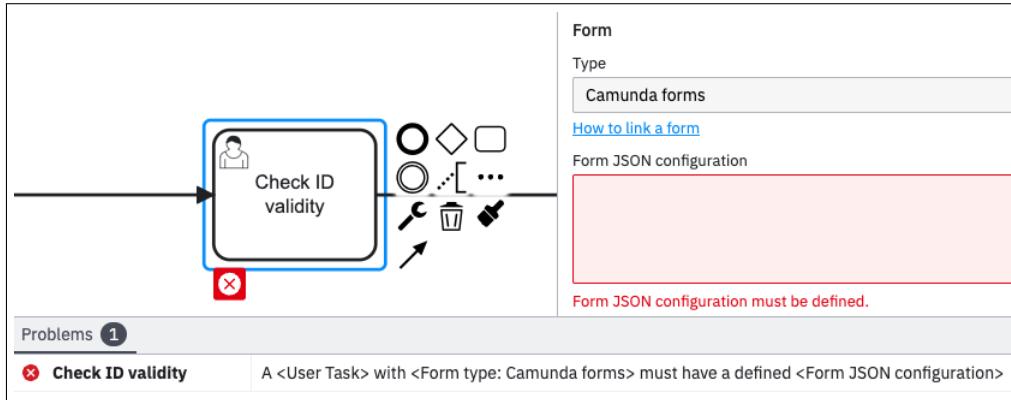


Obr. 5.19: Call Activity 2

### User Task

Úloha slúži na interakciu človeka s procesom. Na jej plnú konfiguráciu je nutná existencia formuláru. Tvorbu formulára sa venujeme v ďalšej sekcií č.5.1.5. Ku konfigurácii tejto úlohy sa teda vrátim až danej sekcií.

Kým však nemáme vytvorený formulár je vhodné v nastaveniach úlohy vybrať v záložke *Form typ Camunda forms*, aby nám nástroj zobrazoval chybovú správu ako pripomienku na jeho vytvorenie.



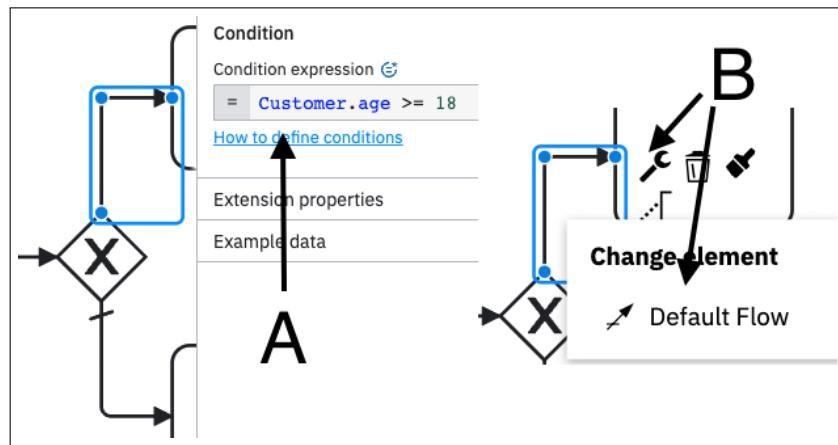
Obr. 5.20: User Task - chybajúci formulár

### *Parallel gateway (AND)*

Nie je potrebná žiadna konfigurácia.

### *Exclusive gateway (XOR) / Inclusive gateway (OR)*

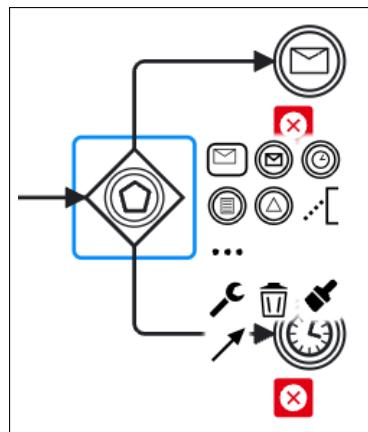
Je potrebné naplniť podmienky v šípkach (A), ktoré z brány vychádzajú. Prípadne nastaviť cestu ako predvolenú (B).



Obr. 5.21: Exclusive gateway (XOR) / Inclusive gateway (OR)

### *Event based gateway*

Z brány môžu vychádzať len udalosti, na ktoré brána čaká. Konfigurovať je teda potrebné iba dané udalosti (vysvetlené v sekcii 5.1.4 krok A.).



Obr. 5.22: Event based gateway

### *Complex gateway*

Nie je podporovaná v aktuálnej verzii (8.2).

### 3. modelovanie objektov

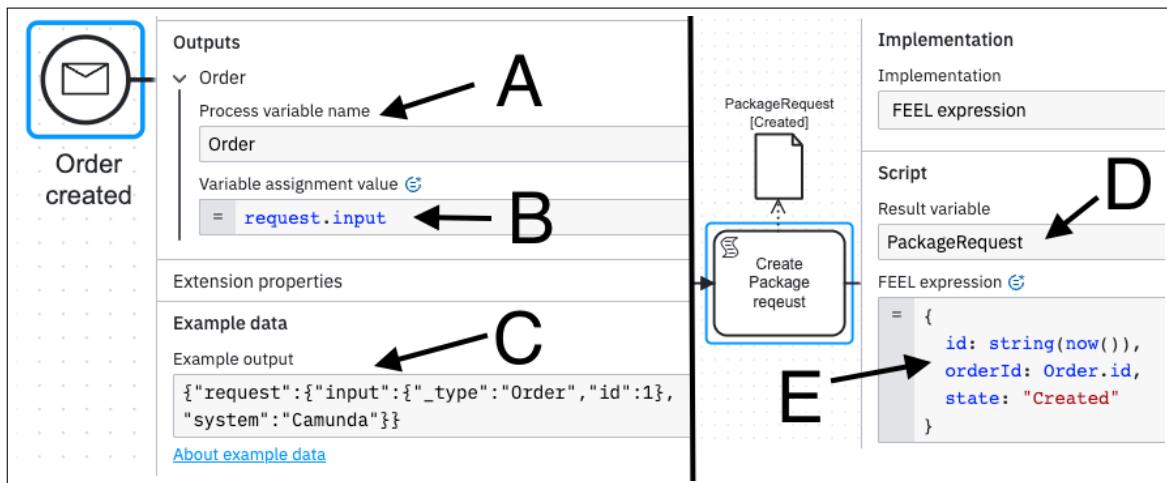
Napriek tomu, že Camunda umožňuje prácu s objektami, v diagrame ma element *Data Object Reference* čisto vizuálny charakter. Doporučene ich do diagramu napriek tomu doplniť nad úlohu, ktorá naozaj mení stav objektu.

#### C. konfigurácia objektov a zmeny ich stavu

Kedže element *Data Object Reference* nemá funkčný charakter, budeme hovoriť o tom ako vytvoriť objekt a meniť jeho stav. Existuje viacero spôsobov ako meniť stav objektov.

##### Vytvorenie objektu

Ak vstupuje objekt do procesu zo správy alebo úlohy, je potrebné tento objekt premapovať. V nastaveniach elementu definujeme názov lokálnej premennej v procese (A), pod ktorým bude dostupný a názov premennej pod ktorou ho v prichádzajúcich dátach nájdeme (B). V prípade, že správa alebo dátá z úloh majú definovaný nejaký formát dát, je vhodné tento formát označiť do parametra *Example output* (C). Pri mapovaní v bode B nám bude nástroj našepkávať podľa uvedeného formátu.



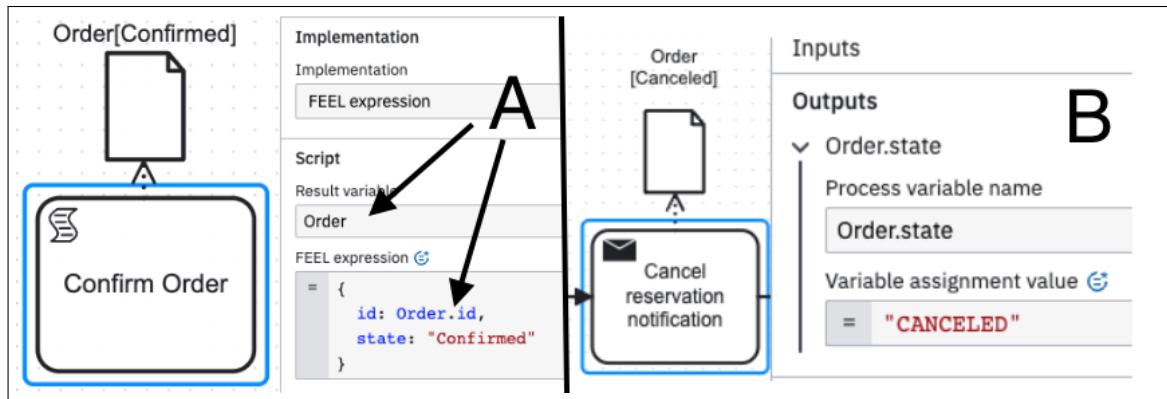
Obr. 5.23: Vytvorenie objektu

Ak chceme objekt vytvoriť v procese, použijeme *Script task*. V záložke *Script* definujeme názov výstupnej premennej (D) a do *FEEL expression* uvedieme pomocou JSON podobu objektu (E).

### Zmena stavu objektu

Najvhodnejšou variantou je zmenu stavu vykonať pomocou *Script task*. Ako názov výstupnej premennej uvedieme používaný názov premennej, ktorá drží objekt (A). Čím dojde k jej prepísaniu. Do *FEEL expression* musíme vložiť telo objektu a urobiť premapovanie atribútov, ktoré nechceme meniť (A). Atribútom, ktoré chceme zmeniť (stav objektu) zadáme novú hodnotu.

Druhou možnosťou je meniť stav objektu pomocou výstupnej premennej úlohy (B). Tu môžeme pristupovať iba na jednotlivé atribúty objektu, a zmeniť tak iba atribút stavu, bez potreby premapovania ostatných atribútov.

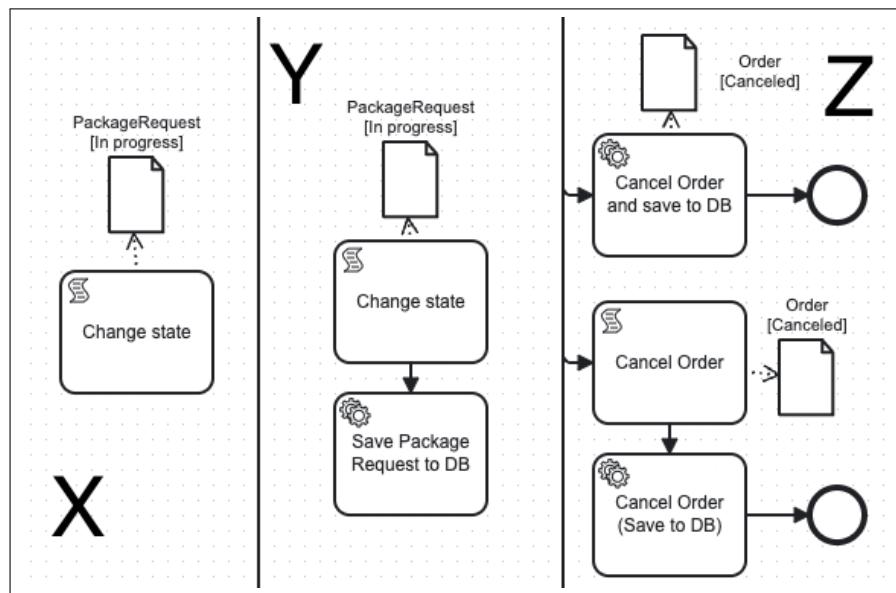


Obr. 5.24: Zmena stavu objektu

Poslednou možnosťou zmeny stavu je v rámci služby (*Service Task*). Hodnota nového stavu môže služba poznať alebo ju pošleme ako vstupnú premennú. Tu záleží natom ako je definované rozhranie služby.

### *Ukladanie stavu*

Je potrebné zvážiť kedy je stav objektu potrebné uložiť do databázy a kedy nám postačuje, že je stav objektu uložený len v rámci inštancie procesu. Pre objektoch, ktoré nemajú dlhý životný cyklus si môžeme vystačiť bez priebežného ukladania do databázy (X). Pri objektov, ktoré majú dlhý životný cyklus alebo strata informácie o ich stave v prípade zlyhanie inštancie procesu môže mať veľký negatívny dopad, je potrebné stav priebežne do databázy ukladať (Y). Pri všetkých objektoch, ktorých stav môže podnik zaujímať aj po ukončení procesu je potom potrebné uložiť stav do DB na konci jeho životného cyklu alebo na konci procesu (Z).



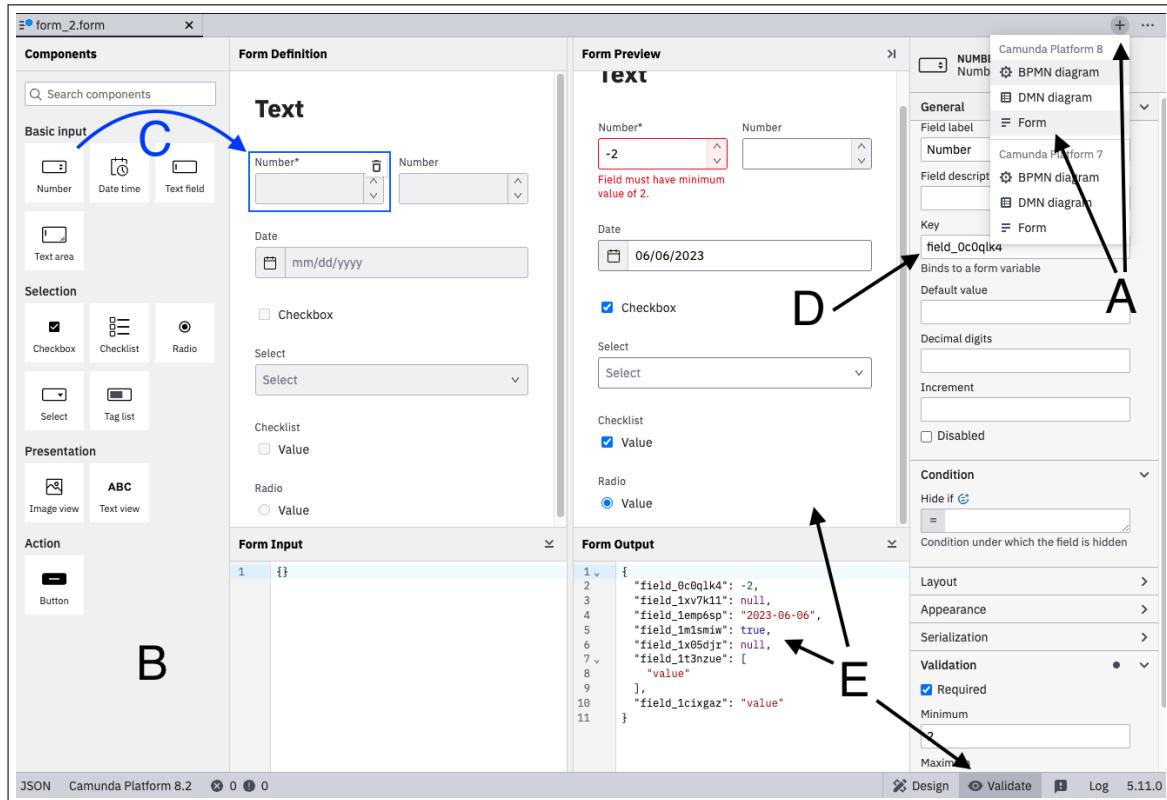
Obr. 5.25: Ukladanie stavu objektu

### 5.1.5 Tvorba formulárov

Tento krok vykonávame na všetky *User Task* elementoch v rámci pracovnej jednotky, ktorá ich obsahuje. Ku jeho vykonaniu je teda potrebné mať transformovaný proces (5.1.4).

#### 1. vytvorenie súboru formulára

Vytvoríme súbor pre formulár (A) a uložíme ho do projektovej štruktúry na doporučené miesto (<project-name>/FORMS/<process-name>/<form-name>).



Obr. 5.26: Tvorba formulárov

#### 2. modelovanie vstupov

Pomocou tabuľky vstupov a výstupov úloh vyhľadáme potrebné formulárové vstupy a následne pomocou diagramu tried alebo vyberieme vhodný typ vstupu. Elementy vyberáme z bočnej ponuky (B) a jednoducho presunieme na modelovaciu obrazovku (C). Nástroj ponúka možnosť skladania vstupov pod seba alebo vedľa seba do stĺpcov.

#### 3. konfigurácia vstupov

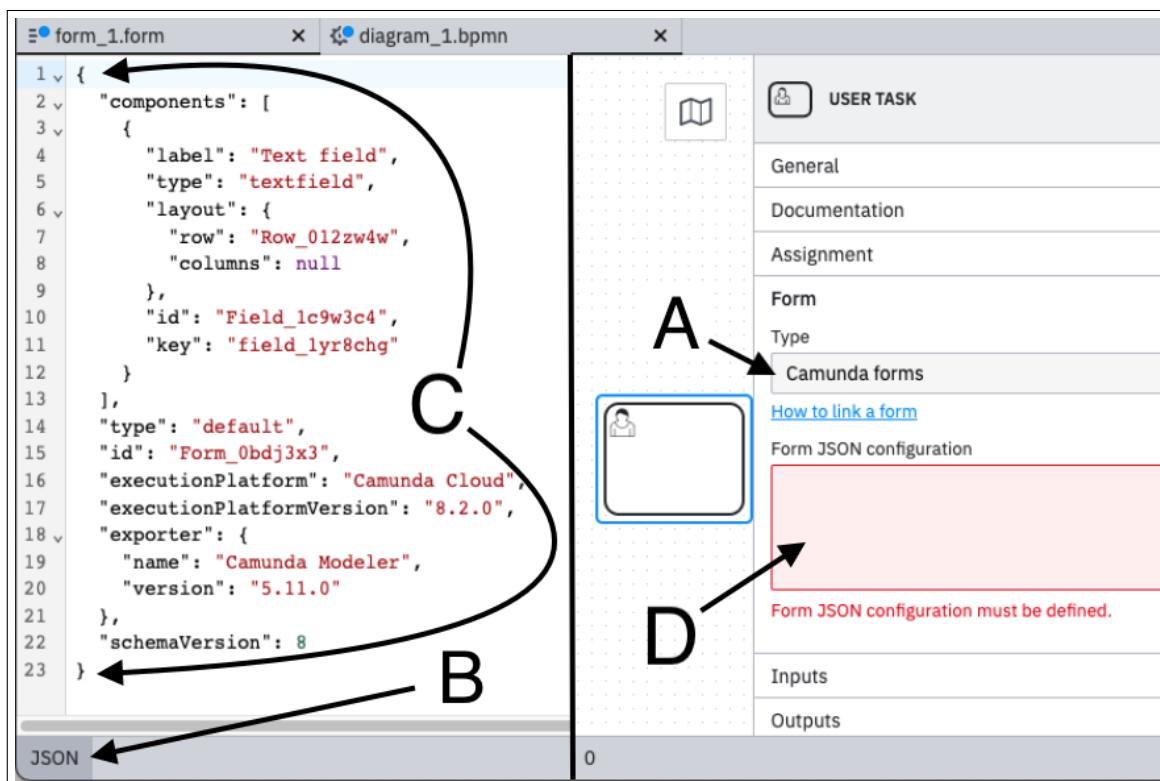
V nastaveniach elementu vstupu je najdôležitejším parametrom *Key* (D), ktorý mapuje výstupnú hodnotu na premennú v procese. Ďalej môžeme podľa rôzneho typu vstupov konfigurovať validačné pravidlá alebo podmienku, pri ktorej splnení bude vstup skrytý. Rozhranie ponúka dva typy zobrazenia. Prvé zobrazuje iba modelovaciu obrazovku.

Druhý typ zobrazenia (E) nám rozdelí obrazovku na dve polovice a na jednej zobrazí náhľad podoby formuláru. Na tejto obrazovke môžeme vyskúšať používanie formuláru a pod obrazovkou vidíme ako sa vstupy formuláru mapujú na výstupy (E).

#### 4. napojenie na proces

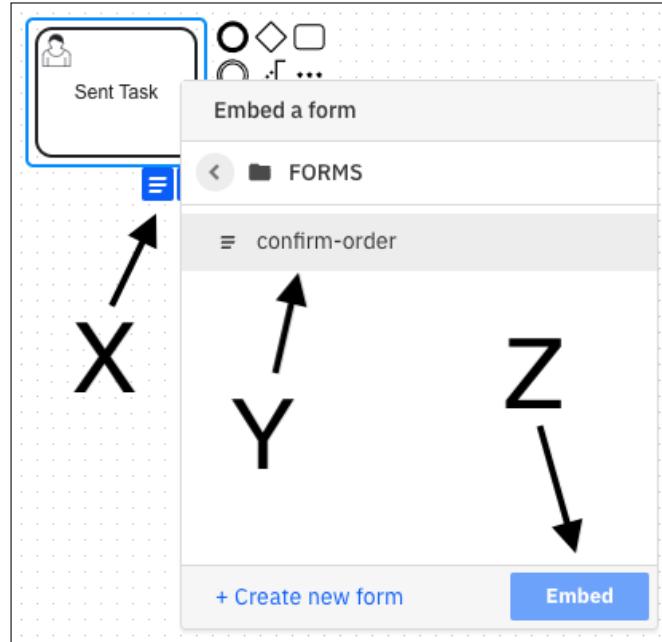
Prepojenie formuláru s procesom je vyriešené dosť nekomfortným spôsobom. Napriek tomu, že *User Task* ponúka možnosť prepojenia cez klíč formulára, prakticky toto prepojenie nefunguje. Je to asi pozostatkom z verziu 7, kde takéto prepojenie funguje. Ostáva dúfať, že v ďalších verziách sa táto možnosť znova sfunkční.

V procese v úlohe do ktorej chceme formulár pridať musíme nastaviť type formuláru *Camunda forms* (A). Vo formulári sa prepneeme do JSON reprezentácie namodelovaného formuláru (B) a skopírujeme celú JSON reprezentáciu (C). Následne ju prilepíme do *User Task* v procese (D).



Obr. 5.27: Napojenie formulára na proces v offline modeléry

Online verzia modeléra tento nekomfortný spôsob aspoň trochu zjednodušuje pomocou. Element ponúka tlačidlo na importovanie (X), ktoré ponúkne plávajúcu ponuku, v ktorej je možné pohybovať sa v projektovej štruktúre. Po vybratí požadované formuláru (Y) a stlačenia tlačidla (Z) sa formulár automaticky vloží.

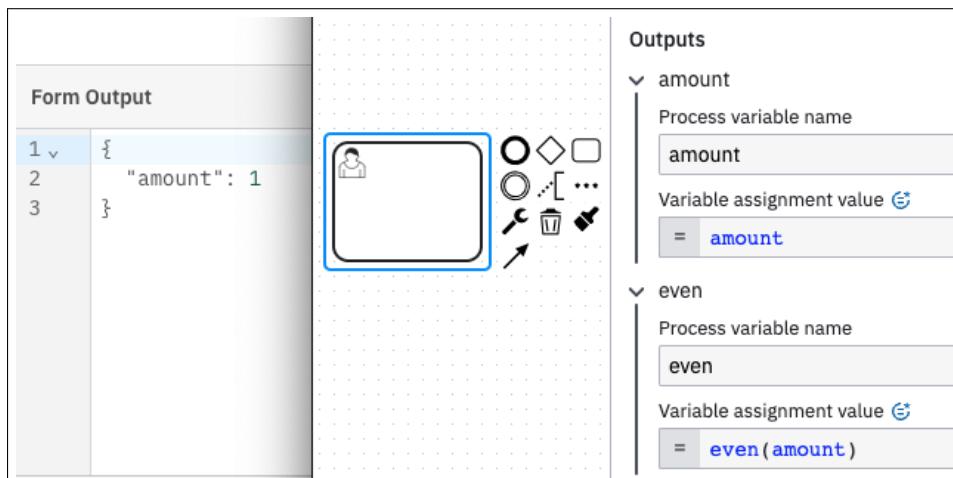


Obr. 5.28: Napojenie formulára na proces v online modeléry

V oboch verziách je tento postup nutné zopakovať ak došlo knejakej zmene vo formulári.

##### 5. mapovanie formulárových vstupov na výstupy úlohy

Ako posledný krok je potrebné namapovať formulárové vstupy (podľa názvu kľúča vstupu) na výstupy z *User Task* úlohy. Môže ísť o obyčajné premapovanie alebo môžeme pridať nové výstupy, ktoré využívajú hodnoty formulárových vstupov.

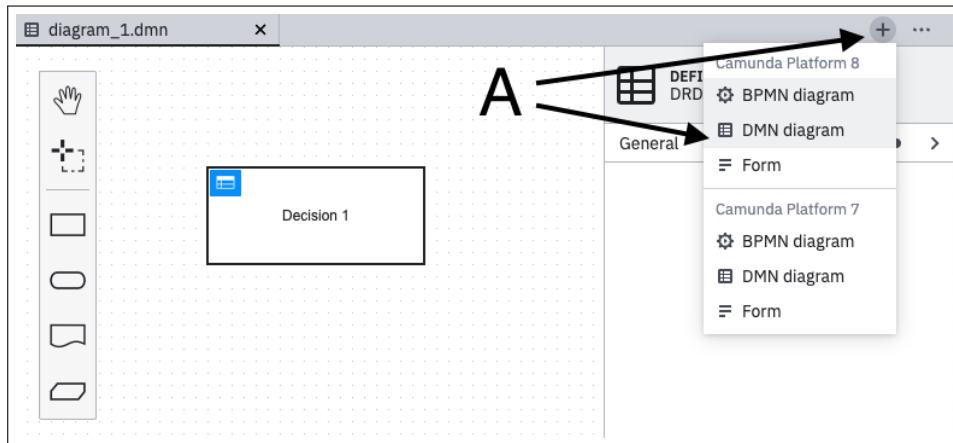


Obr. 5.29: Mapovanie formulárových vstupov na výstupy úlohy

## 5.1.6 Tvorba rozhodovacích diagramov a tabuľiek

### 1. vytvorenie súboru pre rozhodovací diagram

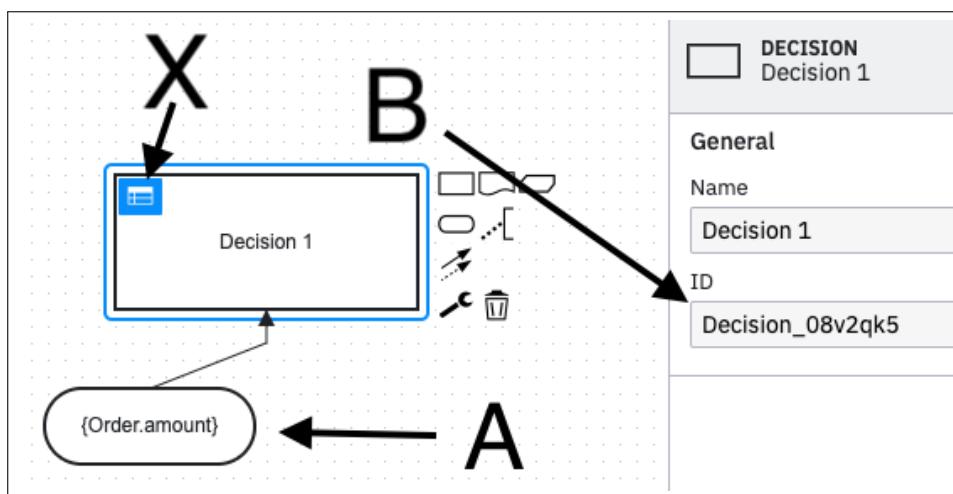
Vytvoríme súbor pre rozhodovací diagram (A) a uložíme ho do projektovej štruktúry na doporučené miesto (<project-name>/DMN/<decision-name>).



Obr. 5.30: Vytvorenie súboru pre rozhodovací diagram

### 2. transformácia DRD diagramu

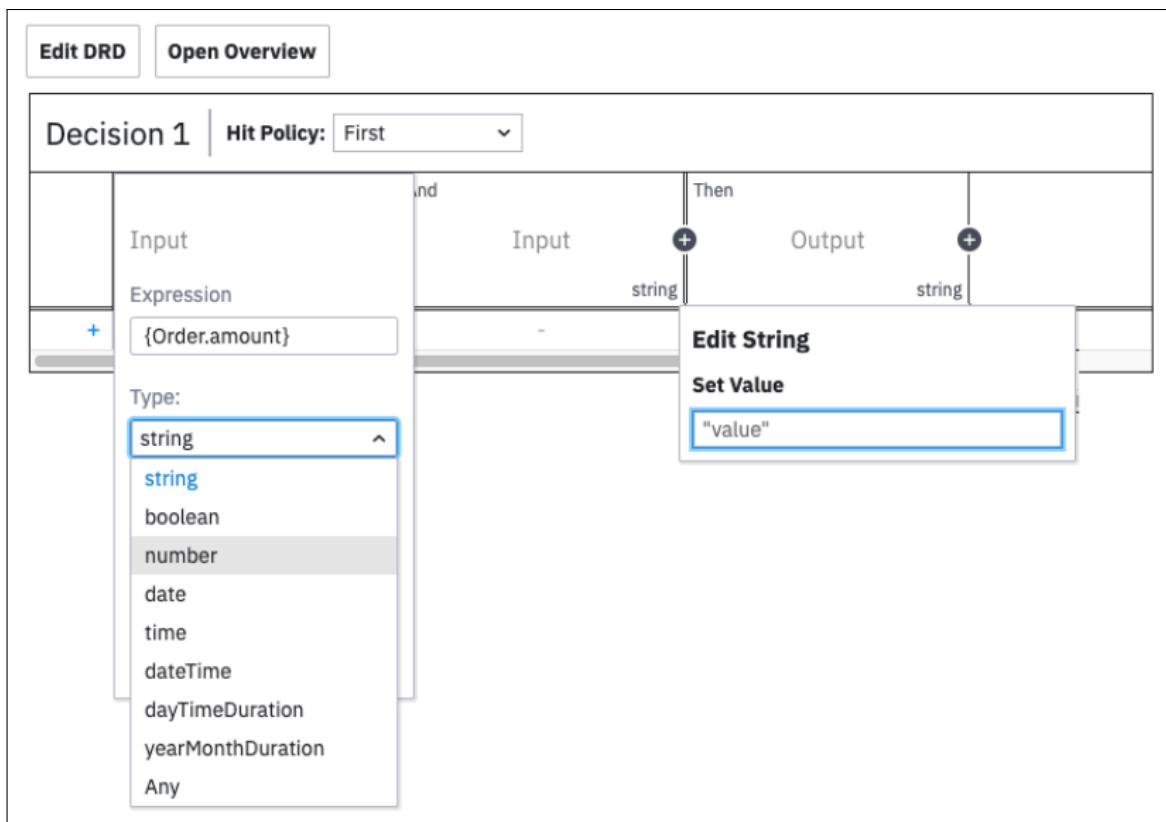
Ak sme sa doteraz držali technologického modelu, tak medzi implementačným a technologickým DRD nemusí byť žiadny rozdiel. Ak ale v implementácii používame iné názvy objektov alebo premenných, zmeníme tieto hodnoty aj v DRD diagrame (A). Paramater *ID* (B) bude slúžiť ako identifikátor v konfigurácii *Business Rule Task*, o ktorej sme hovorili v sekcií 5.1.4.



Obr. 5.31: Implementačný DRD diagram

### 3. naplnenie rozhodovacej tabuľky

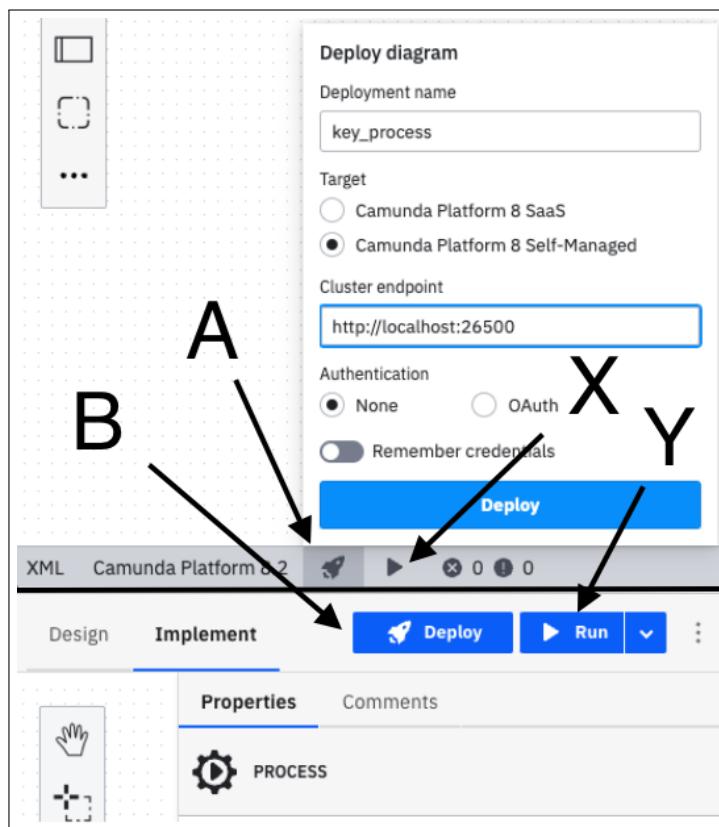
Cez tlačidlo v DRD diagrame (X na obrázku č.5.31) sa presunieme do rozhodovacej tabuľky. Ak bol technologický model rozhodovacej tabuľky správny a pravidla boli napsané v súlade s jazykom FEEL, v tomto kroku iba vkladáme hodnoty do tabuľky. Ak ale v implementácii používame iné názvy objektov alebo premenných, zmeníme tieto hodnoty aj v rozhodovacej tabuľke. Ak technologická tabuľka nepoužívala správne jazyk FEEL alebo zložitejšie výrazy iba popisovala slovne, musíme tieto výrazy upraviť alebo vytvoriť.



Obr. 5.32: Naplnenie rozhodovacej tabuľky

### 5.1.7 Nasadenie diagramov

Predposledným krokom je nasadenie diagramov. Diagramy je možné nasadiť len na bežiaci cluster. Ak používame Self-managed verziu, musí byť spustený Zeebe docker image. V offline verzii modeléra sa nachádza tlačidlo na nasadenie v ľavom dolnom rohu obrazovky (A). V online verzii naopak v pravom hornom rohu obrazovky (B) pri type obrazovky *Implement*.



Obr. 5.33: Nasadenie diagramov

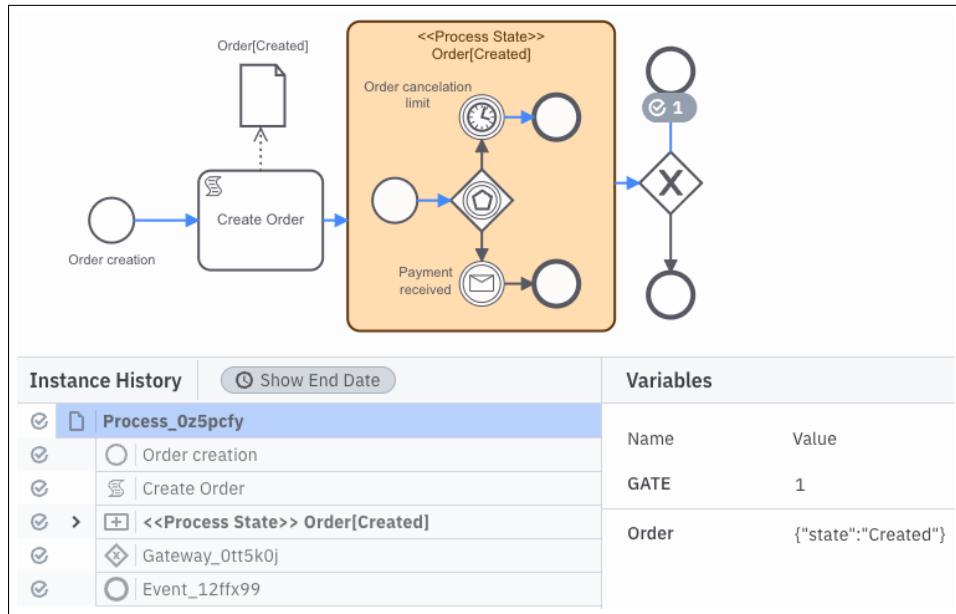
Je potrebné nasadiť procesné diagramy a DRD diagramy.

### 5.1.8 Spustenie diagramov a ich vyladenie

Posledným krokom je spustenie procesu. Ak je proces spúštaný manuálne, tak sa tlačidlo na spustenia nachádza na podobnom mieste ako tlačidlo nasadenia. V offline verzii modeléra v ľavom dolnom rohu obrazovky (X na obrázku č.5.33). V online verzii naopak v pravom hornom rohu obrazovky (Y na obrázku č.5.33) pri type obrazovky *Implement*.

Po spustení je potrebné vyskúsať prechod cez všetky konce pracovnej jednotky. Pomôžeme si prostredím Operate, v ktorom vieme skontrolovať históriu prechodu procesom ale aj aktuálne hodnoty premenných (obrázok č.5.34). Ale aj prostredím Tasklist, v ktorom vykonávame úlohy priradené užívateľom.

Pri SaaS verzii sú všetky rozhrania online a prechod medzi jednotlivými rozhraniami funguje jednoducho cez bočnú ponuku. V Self-managed verzii musia bežať Operate a Tasklist docker images. V prednastavenej konfigurácii sú rozhrania dostupné prostredníctvom internetového prehliadača na nasledujúcich adresách. Operate je dostupný na adresu <http://localhost:8081> a rozhranie Tasklist na adrese <http://localhost:8082>.



Obr. 5.34: Kontrola funkčnosti cez rozhranie Operate

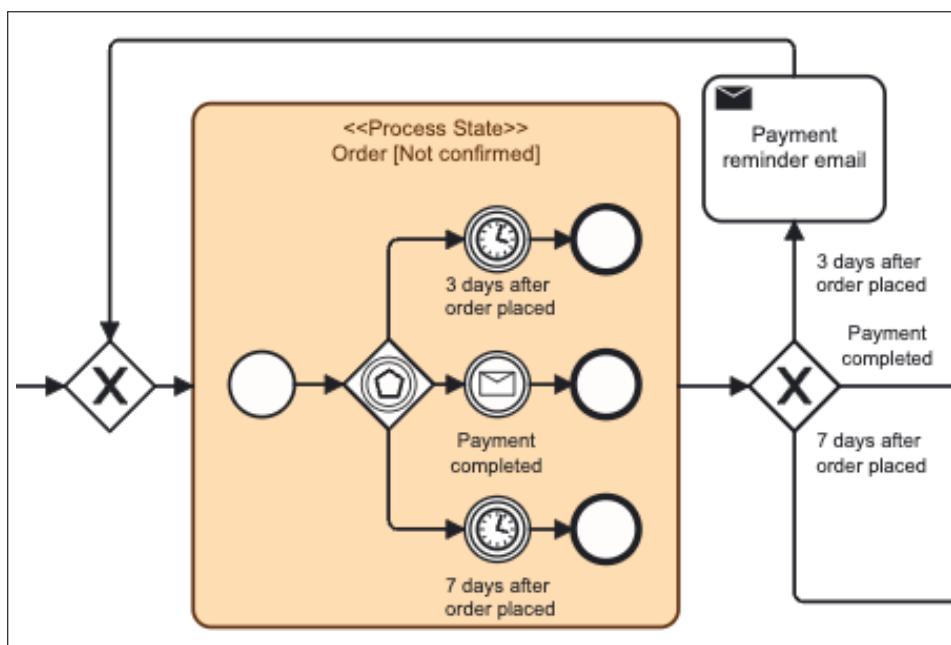
V prípade výskytu problémov pri prechode procesom musíme dané chyby opraviť. Pri bezproblémovom prechode si vyberieme novú pracovnú jednotku a vrátíme sa do bodu 5.1.3. Takto cyklicky pokračujeme až kým nebude hotový celý proces.

### 5.1.9 Problémy a doporučenia

Táto sekcia popisuje objavené problémy pri implemtácii a doporučenia na ich riešenie, či obecné doporučenia, ktoré môžu zjednodušiť vývoj.

#### Dva časové udalosti v stave procesu

**Problém:** V stave procesu môžeme mať dve časové udalosti. Po spustení jedna z nich sa vykoná nejaká činnosť a návrat do rovnakého stavu procesu. V takomto prípade môže dôjsť k zacykleniu a časová udalosť z dlhšou dobou trvania nikdy nenastane (obrázok č.5.35). Časovače sa spúšťajú v okamihu kedy sa flow procesu dostane do *Event Based* brany a vypínajú sa v okamihu ak nastane ktorákoľvek z udalostí. Teda pokial nenastane externá udalosť tak sa proces zacyklí cez jednu časovú bránu.



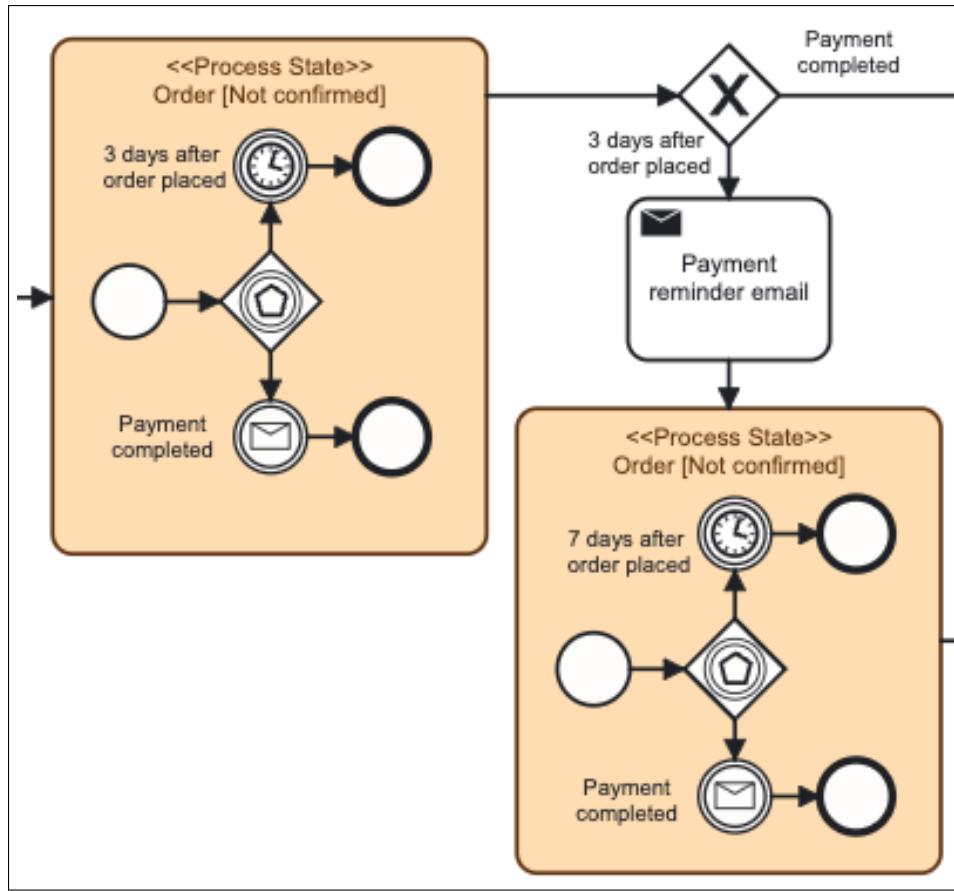
Obr. 5.35: Zacyklenie dvoch časových udalostí

**Riešenie:** Rozdelíme procesný stav pomocou dvoch *Event Based* brán. Každý z nich bude mať jednu z dvoch časových udalostí (obrázok 5.36).

**Nevýhody:**

- Duplicita *Message* udalosti na dvoch miestach.
- Duplicita názvu procesného stavu.

**Výhoda:** Jednoduché riešenie nevyžadujúce obchádzanie pomocou pomocných premenných a zmien hodnôt v časovačoch.



Obr. 5.36: Riešenie zacyklenia dvoch časových udalostí

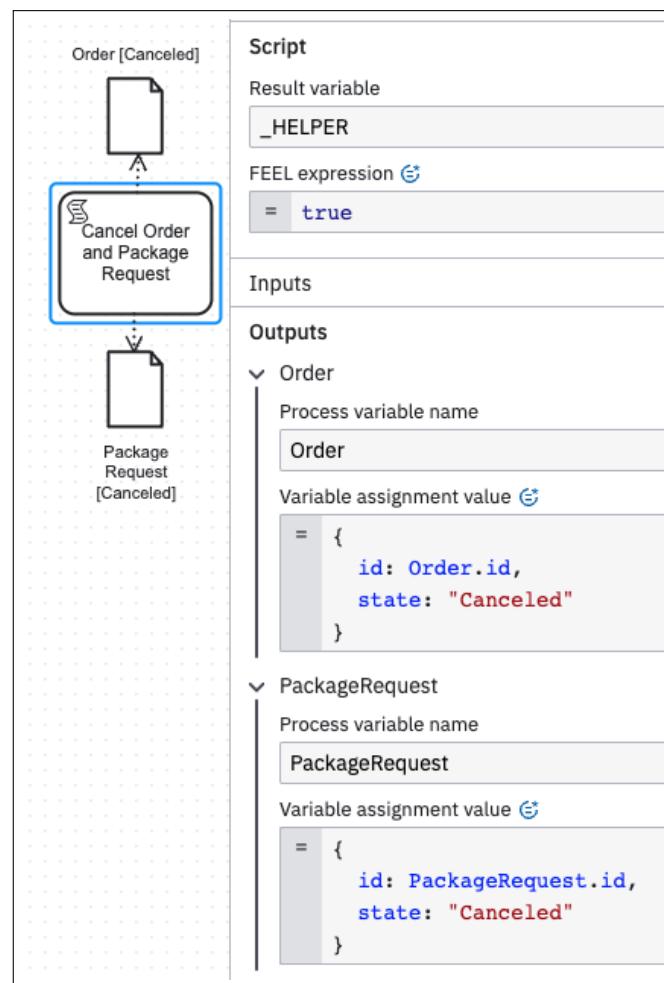
**Alternatívne riešenia:** Existujú aj iné riešenia ale vyžaduje prácu s pomocnými premennými a úpravou hodnôt časovačov za pomocí podmienok. Môže to zároveň vyžadovať pridanie ďalších pomocných elementov.

## Zmena stavu viacerých objektov

**Problém:** Ak meníme stav objektu pomocou *Script Task* tak výstupná premenná skriptu je len jedna. Musíme tak vytvárať *Script Task* pre každý objekt.

**Riešenie:** Namiesto nastavenie stavu objektu v záložke *Skript*, nastavíme výstupné premenné v záložke *Outputs*. Tu môžeme nastaviť hodnoty, pre viacero objektov. Povinné parametre úlohy, ktorými sú *Result variable* a *FEEL expression* naplníme hodnotami, ktoré nebudeme používať (obrázok č.5.37). Pri takomto riešení doporučujeme nezabudnúť pridať element objektu ku danej úlohe, aby bolo jasné, že mení stav viacerých objektov. Zároveň je vhodné "pomocnú" premennú, ktorou obchádzame povinné parametre prepoužívať ak tento postup aplikujeme na viacerých miestach.

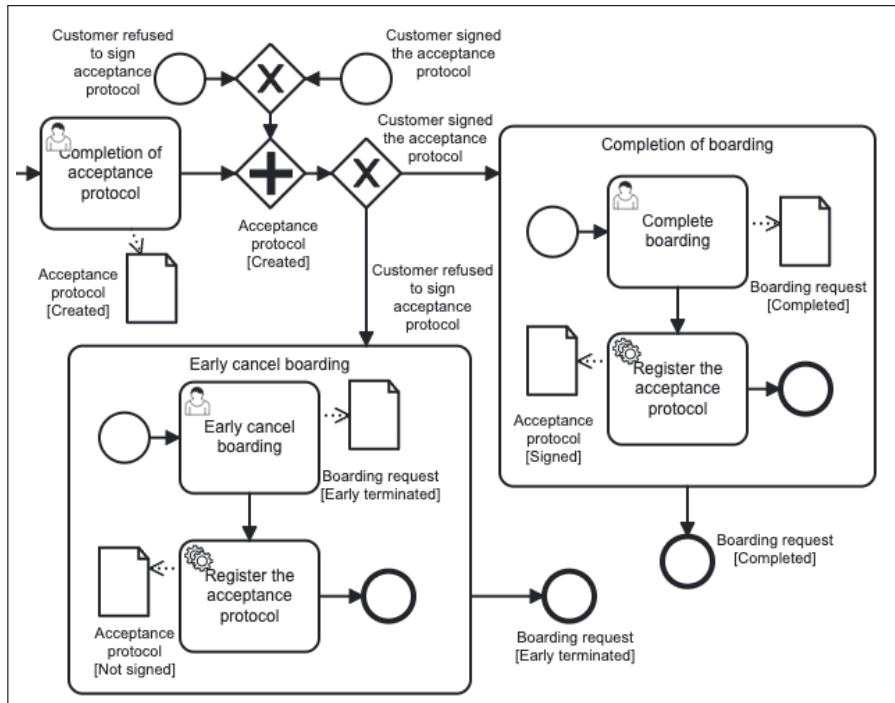
**Nevýhody:** Do procesu sme zanesli zbytočnú premennú.



Obr. 5.37: Nastavenie stavu viacerým objektom pomocou jednej úlohy

## Udalosť z rozhrania Tasklist

**Problém:** Rozhranie Tasklist neponúka možnosť spustenia udalostí. Jediná možnosť ako vytvoriť udalosť je teda prostredníctvom služby a teda nejakého externého rozhrania mimo Tasklist. Existujú však prípady, kedy po dokončení nejakej priradenej úlohy v Tasklistu môže daný užívateľ vyhodnotiť aká udalosť nastala. Praktickým príkladom je vyplnenie nejakého protokolu na základe údajov z úlohy v Tasklistu a následne predloženie zákazníkovi na podpis.

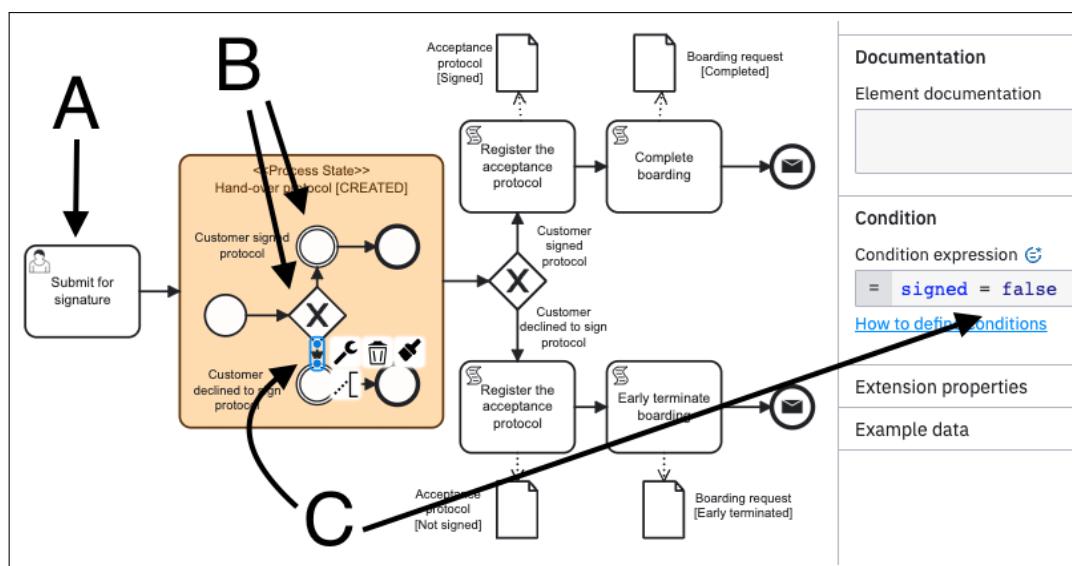


Obr. 5.38: Problém s udalosťou v rozhraní Tasklist

**Riešenie:** Riešenie sa môže zdať priamočiare. Vynecháme procesný stav a nahradíme ho formulárom, kde užívateľ nejakým spôsobom vyberie, ktorá z pôvodných udalostí nastala.

Ak ale chceme zachovať čo najväčšiu konzistenciu medzi technologickým a implementačným modelom, navrhujeme drobnú zmenu v riešení. Namiesto úplného vymazania procesného stavu v ňom iba upravíme jeho elementy.

Pred procesný stav pridáme *User Task* (A) s formulárom, v ktorom budeme zaznamenávať udalosť (napríklad pomocou checkboxu alebo výberu možností). Udalosti, ktoré by pôvodne mali podobu *Message Intermediate Catch Event* nahradíme elementom *Intermediate Throw Event* a *Event Based Gate* nahradíme elementom *Exclusive Gate* (B). Následne nastavíme podmienky na čiarach smerujúcich do udalostí (C). V podmienke použijeme vstup z formulára podľa spôsobu, akým sme implementovali výber udalosti.



Obr. 5.39: Úprava procesného stavu pri vynechaní udalostí

**Nevýhoda:** Komplikovanejšia varianta oproti úplnému odstráneniu procesného stavu.

**Výhoda:** Lepšia konzistencia s technologickými diagramami a zároveň lepší prehľad o stave procesu.

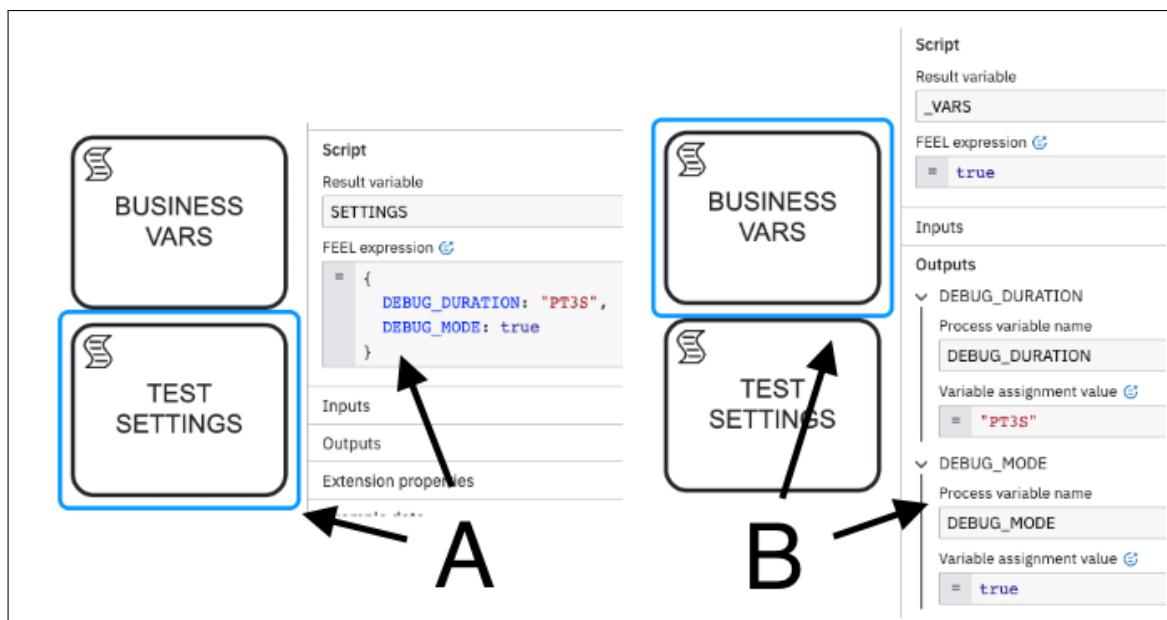
## Testovanie s dlho trvajúcimi časovými odalosťami

**Problém:** Časové udalosti v procese obvykle trvajú dlhšie ako je rozumný čas čakania pri testovaní.

### Riešenie:

1. Vytvoríme pomocnú úlohu typu *Skript Task*, alebo použijeme úlohu v ktorej máme uložené business premenné.
2. Do tela skriptu (A) alebo ako výstupnú premennú zavedieme (B) novú premennú s hodnotou odpovedajúcou dobe trvania časovača pri testovaní (napr. `DEBUG_DURATION="PT5S"`).
3. Do tela skriptu (A) alebo ako výstupnú premennú zavedieme (B) novú pomocnú premennú, ktorá bude slúžiť ako prepínač (napr. `DEBUG_MODE=true`).
4. V časových udalostiach nahradíme dobu trvania nasledujúcou podmienkou:
  - `if <ScriptResultVariable>.DEBUG_MODE then <ScriptResultVariable>.DEBUG_DURATION else <skutocna_doba_trvania>`  
ak sme premenné vkladali do tela skriptu (A)
  - `if DEBUG_MODE then DEBUG_DURATION else <skutocna_doba_trvania>`  
ak sme premenné vytvárali ako výstup z úlohy (B)

Následne meníme hodnotu premennej `DEBUG_MODE` podľa toho, či chceme používať skutočnú dobu trvania alebo testovaciu.



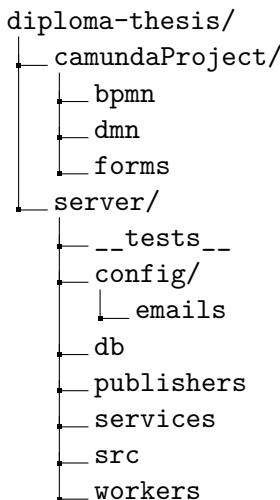
Obr. 5.40: Testovanie s dlho trvajúcimi časovými odalosťami

## 5.2 Overenie metodického postupu transformácie do implementačného modelu nástroja Camunda

Metodicky postup aplikujeme na technologickom modely z kapitoly 4.2. Ten sa skladá z klúčové procesu prenájmu námorného rekreačného plavidla a podporného procesu nalodenia. Vzhľadom nato, že veľká časť metodického postupu tvorí cyklické aplikovanie na menšie pracovné jednotky (napr. transformácia procesu a tvorba formulárov), nebudeme popisovať vývoj úplne každej pracovných jednotiek. Síce sme metodický postup aplikovali na celý proces ale pre popis overenia si vystačíme s vybranými pracovnými jednotkami, na ktorých môžeme ukázať jednotlivé kroky.

### 5.2.1 Vytvorenie štruktúry projektu

Celý náš projekt sa skladá z dvoch častí. Prvú časť tvoria diagramy a formuláre pre Camundu. Keďže Camunda neumožňuje vytváranie udalostí z prostredia Tasklist, tak bolo potrebné vytvoriť nejaké ďalšie rozhranie. Preto bola vytvorená jednoduchá webová stránka, ktorá simuluje podnikovú aplikáciu. Poskytuje rozhranie na vytváranie udalostí a zároveň spravuje služby používané procesom. Táto aplikácia predstavuje druhú časť projektovej štruktúry, ktorú sme nazvali server.



Na obrázkoch č.5.41 a č.5.42 vidíme rozhrania webovej stránky. Hlavná obrazovka ponúka odkazy na zadávanie udalostí. Zároveň sa v spodnej časti nachádza farebne odlišené testovacie/ladiace prostredie. V ňom môžeme spúštať testovacie/ladiace skripty, ktoré spúštajú procesné udalosti v istom poradí, čím simuluje priebeh procesu. Ukážku kódu takého skriptu môžeme vidieť na výpise kódu č.5.1.

# Yacht charter

## Customer

- [Create order](#)
- [Pay deposit](#)
- [Pay order](#)
- [Boarding request](#)
- [Landing request](#)
- [Report incident](#)

## Employee

- [Boarding request](#)
- [Landing request](#)
- [Report incident](#)

Only for testing purpose

### Simulate process

- [Order created](#)
- [Order created, deposit paid](#)
- [Order created, deposit paid, full price paid](#)
- [Order created, deposit paid, full price paid, boarding requested](#)
- [Boarding requested, boarding finished](#)
- [Boarding requested, early terminated](#)

### Events

Event  Correlation key (OrderID)

Obr. 5.41: Implementation: Web interface

## Order create

### Name

Milan Moreplavec

### Email

milan.moreplavec@email.com

### Phone number

777123456

### Start date

dd/mm/yyyy

### End date

dd/mm/yyyy

### Select boat

Bavaria 38 Cruiser

Only for testing purpose

\*Customer credibility (0 - 100)

0

\*Boat name

Neptun 1

\*Capacity (2 - 10)

6

Obr. 5.42: Implementation: Web interface for order creation

```

1 const orderPublisher = require('./publishers/order-publisher');
2 const TIMEOUT = 2500; //2.5 seconds
3 module.exports = {
4     depositPaid: () => {
5         const orderId = orderPublisher.placeNewOrder(mockOrder());
6         setTimeout(()=>{
7             orderPublisher.depositPaymentCompleted(orderId)}, TIMEOUT)
8
9     },
10    fullPricePaid: () => {
11        const orderId = orderPublisher.placeNewOrder(mockOrder());
12        setTimeout(()=>{
13            orderPublisher.depositPaymentCompleted(orderId)
14            setTimeout(()=>{
15                orderPublisher.fullPaymentCompleted(orderId)}, TIMEOUT)
16            }, TIMEOUT)
17 }

```

Výpis 5.1: Debugging skripts

Webová stránka bola vytvorená pomocou frameworku Express. Komunikáciu s Camundou zabezpečuje komunitná verzie Zebee klienta pre NodeJS. Krátku ukážku kódu serveru vidíme na výpise č.5.2.

```

1 const orderService = require('./services/order-service');
2 const orderPublisher = require('./publishers/order-publisher');
3 const orderWorkers = require('./workers/order-workers');
4 const { ZBClient } = require('zeebe-node')
5 const zbc = new ZBClient('localhost', { loglevel: 'ERROR' });
6 /** Workers */
7 zbc.createWorker({
8     taskType: 'order', taskHandler: orderWorkers.order,
9 });
10 /** HTTP Server */
11 const express = require("express");
12 const app = express();
13 app.listen(3000, () => {
14     console.log("Application started and listening on port 3000");
15 });
16 /** Router */
17 app.get("/customer/pay-order", (req, res) => {
18     res.sendFile(filePath(__dirname + "/src/orderPay.html"));
19 });
20 app.post("/customer/pay-order", (req, res) => {
21     orderPublisher.fullPaymentCompleted(req.body.order_number);
22     res.redirect("/");
23 });

```

Výpis 5.2: Web application

### 5.2.2 Výber pracovnej jednotky

Pri aplikovaní metodického postupu sme najprv postupovali podľa doporučenej veľkosti pracovnej jednotky, a teda jeden procesný stav. Pri modelovaní podporného procesu sme vyskúšali pracovnú jednotku zväčšíť na dva procesné stavy. Vzhľadom na väčšie množstvo formulárov v podpornom procese sa ale tento krok ukázal ako chybný a vývoj mierne spomalil.

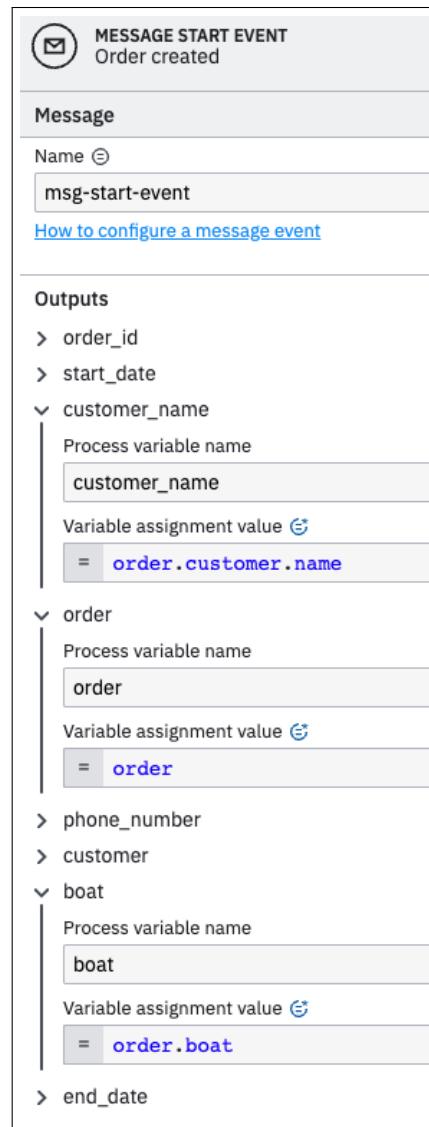
### 5.2.3 Vytvorenie procesného diagramu

#### Vytvorenie BPMN diagramu a pridanie plavebných dráh

Boli vytvorené dva procesné diagramy. Do diagramu pre klúčový proces s názvom súboru `key_process.bpmn` bola vložená jedna plavebná dráha pre rolu `admin staff`. Do podporného procesu `Boarding` s názvom súboru `boarding_support_process.bpmn` boli vložené dve plavebné dráhy pre role `admin staff` a `technical staff`.

#### Konfigurácia vstupnej udalosti

Spúšťacia udalosť klúčového procesu je typu *Message Start Event*. Proces teda spúšta udalosť vytvorenia objednávky. Tá môže vzniknúť z užívateľského rozhrania (obrázok č.5.42) alebo z testovacieho skriptu. Na elemente spúšťacej udalosti bolo potrebné nastaviť názov udalosti a premapovať očakávané vstupy z udalosti do procesu (obrázok č.5.43). Zároveň boli vytvorené aj pomocné premenné za účelom zjednodušenie prístupu k atribútom vybraných objektov (napr. `customer_name = order.customer.name`).



Obr. 5.43: Implementation: Inputs mapping in start event

## Pridanie zoznamu premenných a konštánt

Ako druhý element procesného diagramu bol pridaný *Script Task* s názvom **CONSTANTS**. Do tejto úlohy boli postupne pridávané business premenné ale aj premenné či konštanty technického charakteru (obrázok č.5.44). Využili sme spôsob cez pomocný objekt, ktorý je výsledok skriptu. Zároveň sme vyskúšali aj druhý spôsob, teda nastavenia výstupných premenných z úlohy. Do objektu sme vložili časové známky, business premenné a prepínač ladiaceho režimu procesu. Ako výstup z úlohy používame koštanty predstavujúce názvy stavov. Zároveň bolo nutné premapovať aj pomocný objekt zo skriptu s názvom **VARS**.

```

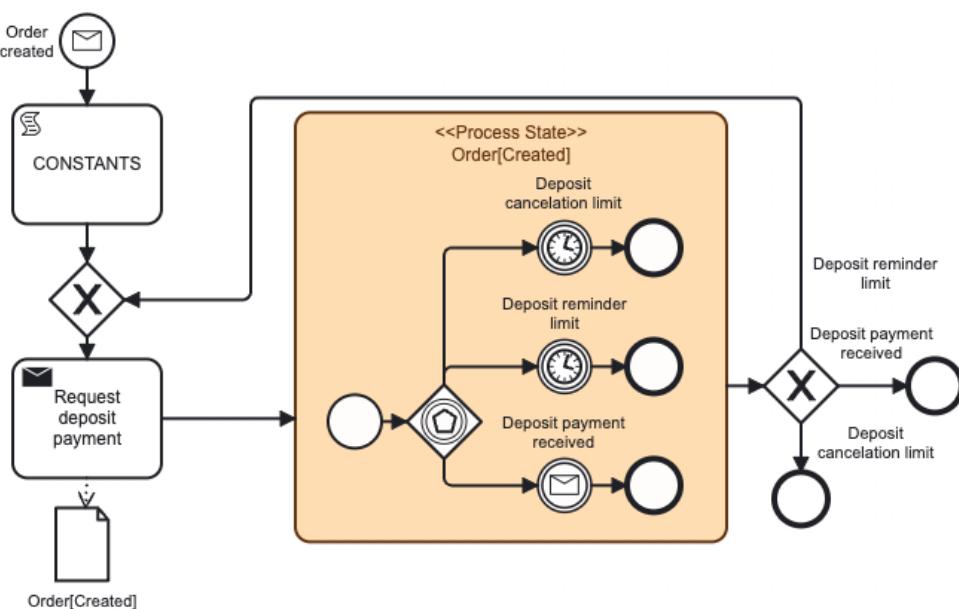
Script
Result variable
VARS
FEEL expression ⓘ
= `

    MODE_DEBUG:true,
    TIME:{ 
        timestamp_debug: "PT3S",
        timestamp_debug_long: "PT30S",
        timestamp_PT6H: "PT6H",
        timestamp_P3D: "P3D",
        timestamp_P4D: "P4D",
        timestamp_P90D: "P90D",
        timestamp_PT30M: "PT30M"
    },
    EMAILS:{ 
        boardingStartTime: "8:00",
        boardingEndTime: "16:00",
        boardRefundPercentage: "20%"
    }
`
```

<b>Inputs</b>	+
<b>Outputs</b>	+
> VARS	
> CREATED	
> ON_CRUISE	
> CONFIRMED	

### 5.2.4 Transformácia procesu

Pozrieme sa postup pri prvej pracovnej jednotke, ktorej podobu vidíme na obrázku č.5.45.



Obr. 5.45: Implementation: first work unit

Po **modelovaní procesného stavu** sme pokračovali na **konfiguráciu výstupu zo stavu a stavových udalostí**. Dobra trvania časových udalostí je uvedená v dňoch. Preto sme zvolili konfiguráciu cez testovacie/ladiace časové známky. Konfigurácia doby trvania časových udalostí tak vyzerala nasledovne:

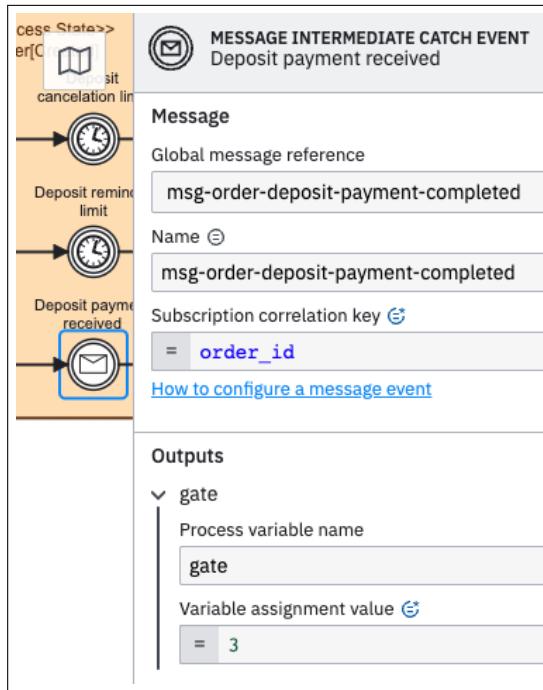
- Deposit cancellation limit

```
if VARS.MODE_DEBUG then VARS.TIME.timestamp_debug_long
else VARS.TIME.timestamp_P4D
```

- Deposit reminder limit

```
if VARS.MODE_DEBUG then VARS.TIME.timestamp_debug
else VARS.TIME.timestamp_P3D
```

Všetkým udalostiam bola nastavená výstupná premenná **gate** s hodnotou poradia udalosti. Následne bolo do ciest z XOR brány za stavom pridaná podmienky pre porovnávanie tejto premennej. Udalosti prijatie platby bol nastavený názov správy a korelačný kľúč (obrázok č.5.46).



Obr. 5.46: Implementation: message event configuration

Po **modelovaní úloh a brán** bolo potrebné **konfigurovať úlohy a brány**. Okrem brány vychádzajúcej z procesného stavu máme v tejto pracovnej jednotke len jednu XOR bránu, ktorá má charakter spájanie ciest a nie je potrebná jej konfigurácia. Nachádza sa tu jedna úloha typu *Send Task*. Vzhľadom nato, že Self-managed verzia neobsahuje konektor na externú emailovú službu, tak implementácia odosielanie emailov ostáva na nás. Kedže takáto implementácia nie je cielom práce, emailsy nebudem naozaj posielat ale budeme iba vypisovať ich podobu v konzolovom okne serveru služby (obrázok č.5.47).

```

email-worker::[1687529784364]
-----::email[email1687529784364@email.com]

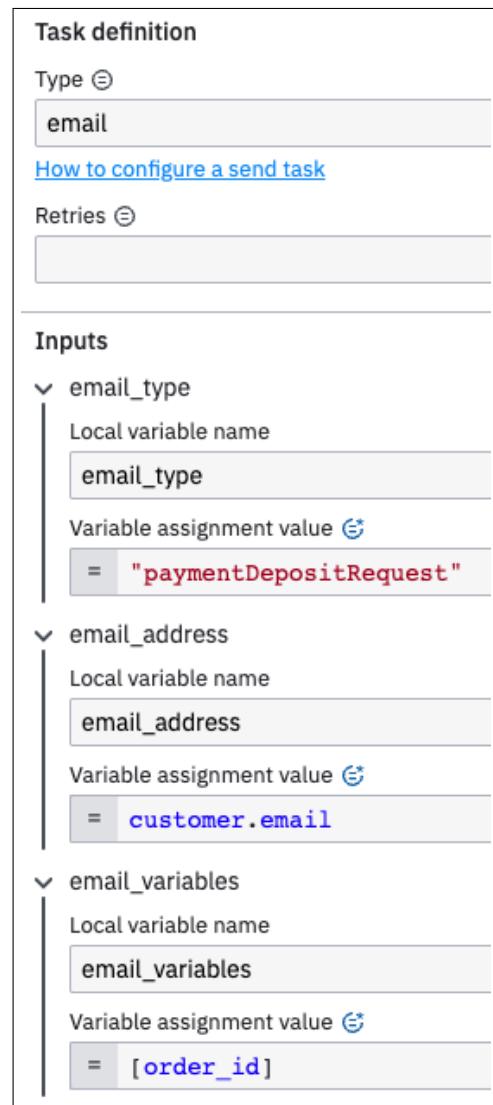
Boarding date for order n.'1687529784364'
has been changed to 'Thu Jun 27 2024'.
Arrival for boarding is required from '8:00' to '16:00'.

The Charter company
-----
```

Obr. 5.47: Implementation: Email log

Nevyhnutné bolo ale navrhnúť spôsob, akým sa vymieňajú dátá medzi procesom a službou, ktorá ma emaily posielat (v našom prípade vypisovať do konzoly). Na servery sme tak vytvorili emailové šablóny (ukážka vo výpise kódu č.5.3) v JSON formáte podľa zoznamu notifikáciu. V prípade zmeny emailovej šablóny tak máme k dispozícii konfiguračný súbor, ktorý môžeme kedykoľvek meniť a nie je potrebné znova nasadzovať server.

Úlohy v procese potom majú 2 alebo 3 vstupy. Názov emailovej šablóny v premennej `email_type` a adresu odosielateľa v premennej `email_address`. Ak šablóna obsahuje premenné tak úloha posielá vstup pod názvom `email_variables`. Hodnotu vstupu tvorí pole s hodnotami premenných v zhodnom poradí ako vidíme na výpise č.5.3 v poli `_variables` pre konkrétnu šablónu. Ukážku takejto konfigurácie vidíme na obrázku č.5.48.



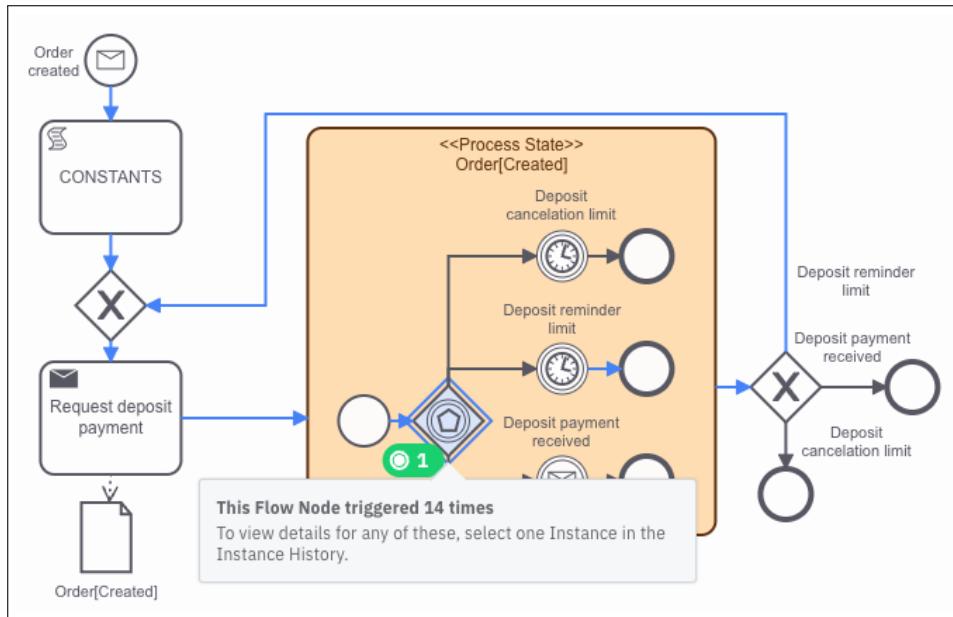
Obr. 5.48: Implementation: Email configuration

```

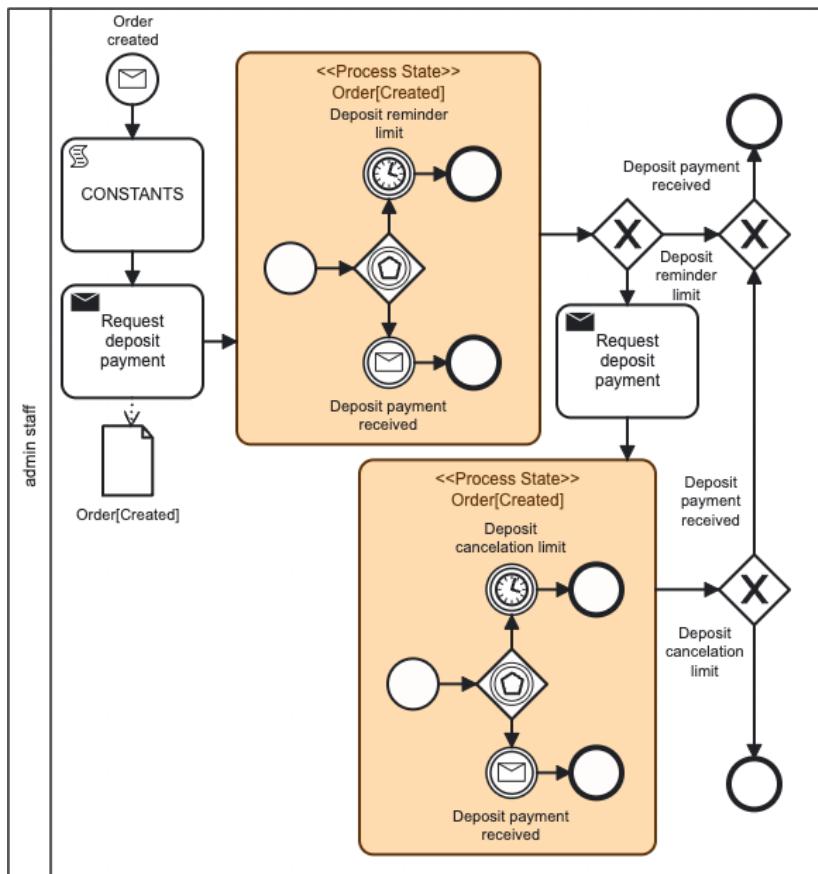
1 {
2   "paymentDepositRequest": {
3     "text": "We remind you that\nthe deposit for the order n.%o\nmust be
4     paid before the 7th day from order creation.\n\nThe Charter company",
5     "_variables": ["orderNumber"] },
6   "paymentFullRequest": {
7     "text": "We remind you that\nfull price (%o EUR) for the order n.%o\
8     nmust be paid before the 7th day from order creation.\n\nThe Charter
9     company",
10    "_variables": ["price", "orderNumber"] },
11   "cancelReservation": {
12     "text": "Unfortunately, I have to inform you\nthat we canceled your
13     order n.%o\nbecause you did not pay the deposit.\n\nThe Charter company",
14     "_variables": ["orderNumber"] },
15   "cancelOrderNoRefund": {
16     "text": "Unfortunately, I have to inform you\nthat we canceled your
17     order n.%o without refundation.\nFor further information, contact\nthe
18     customer department at %o.\n\nThe Charter company",
19     "_variables": ["orderNumber", "infoPhoneNumber"] },
20   "orderCompleted": {
21     "text": "Your order n.%o has been paid and completed.\nWe look forward
22     to your arrival on %o\nArrival for boarding is required from %o to %o.\n\n
23     The Charter company",
24     "_variables": ["orderNumber", "cruiseStartAt", "boardingStartTime", " "
25       "boardingEndTime"] },
26   "changedBoardingDate": {
27     "text": "Boarding date for order n.%o\nhas been changed to %o.\n
28     nArrival for boarding is required from %o to %o.\n\nThe Charter company",
29     "_variables": ["orderNumber", "movedBoardingDate", "boardingStartTime",
30       "boardingEndTime"] },
31   "orderCanceledPenalty": {
32     "text": "We are sorry, but your order n.%o has been exceptionally
33     cancelled.\nWe apologize for the caused problems. You will be refunded
34     full price plus\n%o of the rental price and you are entitled to adequate\
35     nfinancial compensation according to the general business condition.\n\n
36     The Charter company",
37     "_variables": ["orderNumber", "boardRefundPercentage"] },
38   "orderExceptionallyTerminated": {
39     "text": "We apologize that we could not make the planned landing.\nYou
40     will be contacted by phone regarding further information.\nWe apologize
41     for the caused problems. You will be refunded full price plus\n%
42     landRefundPercentage of the rental price and you are entitled to adequate\
43     ncompensation according to the general business condition.\n\n
44     The Charter company" },
45   "orderFinish": {
46     "text": "Your rental (order n.%o) has been successfully completed.\nWe
47     look forward to your next cruise on one of our ships.\n\n\nThe Charter
48     company",
49     "_variables": ["orderNumber"] }
50 }
```

Výpis 5.3: Email templates

Po nasadení tejto pracovnej jednotky sme zistili, že došlo k zacykleniu (obrázok č.5.49).  
 Táto jednotka teda podnietila vznik sekcie Problémy a doporučenia v metodickom postupe.  
 Problém sme vyriešili rozpadom na dva brány (obrázok č.5.50).

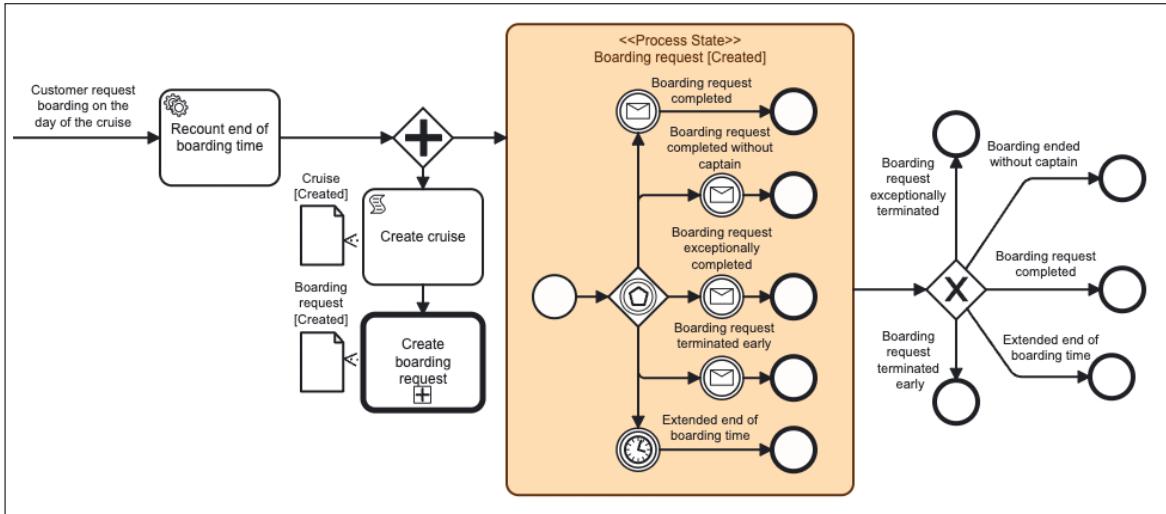


Obr. 5.49: Implementation: timer event cycle



Obr. 5.50: Implementation: first work unit

Na ďalšej pracovnej jednotke, ktorú vidíme na obrázku č.5.51 si ukážeme konfiguráciu *Service Task*, *Script Task* a volania podporného procesu.



Obr. 5.51: Implementation: work unit with service task, script task and support process call

Pre prácu so službami bolo vytvorených niekoľko tzv. *job workers*. Server, ktorý poskytuje užívateľskú webovú stránku sa zároveň stará o všetky služby. V dokumentácii (ukážka vo výpise kódu č.5.4) pracovníkov sa nachádzajú povinné vstupy. Vo výpise č.5.5 vidíme ukážku dokumentácie konkrétnej funkcie s povinnými vstupmi a očakávaným výstupom, prípadne chybou. Na základe týchto údajov môžeme konfigurovať jednotlivé servisné úlohy.

```

1  /** Email worker * @type 'email'
2   * @input email_type -- name of service
3   * @input email_adress -- receiver email address */
4 zbc.createWorker({ taskType: 'email', taskHandler: emailWorkers.email });
5  /** Order worker * @type 'order'
6   * @input operation_type -- name of service */
7 zbc.createWorker({ taskType: 'order', taskHandler: orderWorkers.order });
8  /** Timestamp worker * @type 'timestamp'
9   * @input operation_type -- name of service */
10 zbc.createWorker({
11   taskType: 'timestamp', taskHandler: timestampWorkers.timestamp });

```

Výpis 5.4: Job workers

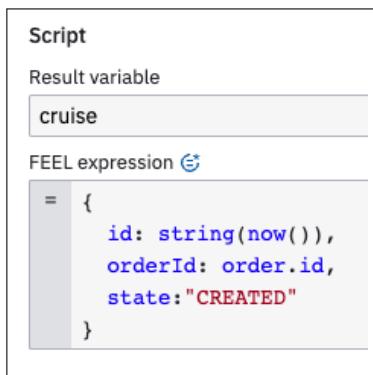
```

1  /** Reschedule days * type = "reschedule-days"
2   * input (optional) = max_days
3   * input = reschedule_days -- number of days to reschedule
4   * input = end_hours -- end of time at rescheduled day
5   * input = original_date -- original date
6   * input = date -- actual date (could be rescheduled already)
7   * output = date
8   * output = end_time -- duration to rescheduled date in ISO8601
9   * throw = 4002 -- when reschedule_days is more than max_days
10  * or difference between original_date and new date is more than max_days */

```

Výpis 5.5: Function documentation

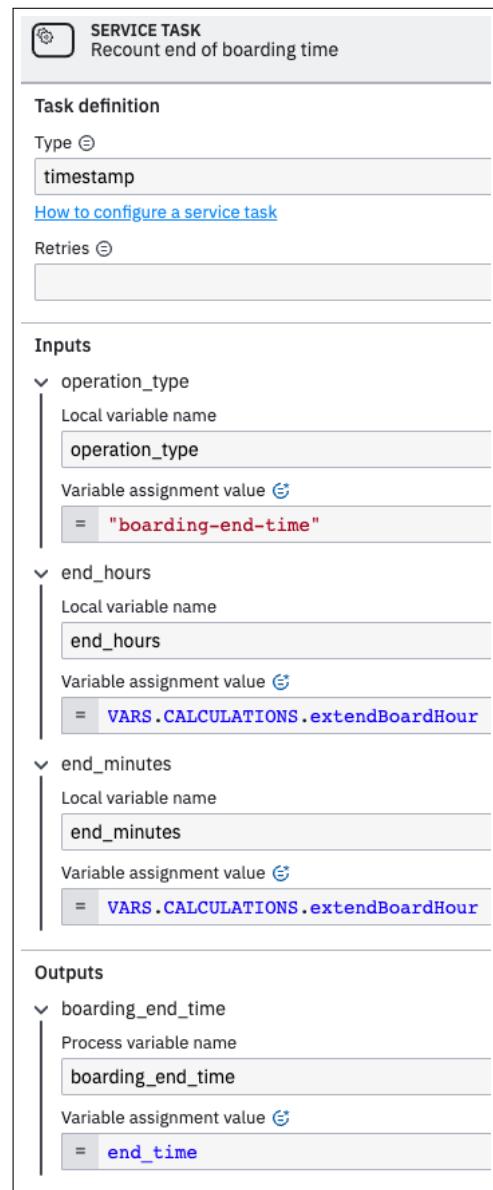
Popíšeme si teda jednotlivé časti obrázku č.5.51. Konfiguráciu služby na výpočet doby trvania, používanej v procesnom stave, vidíme na obrázku č.5.52. Za touto službou sa pomocou *Script Task* vytvára nový objekt (obrázok č.5.53). Následne je volaný podporný proces pomocou *Call activity*, v ktorej sa vytvára nový objekt (obrázok č.5.54) pomocou vstupu do podporného procesu. Zároveň je povolená propagácia premenných aby sme využili všetky výhody tohto elementu.



Obr. 5.53: Implementation: Script task creating object

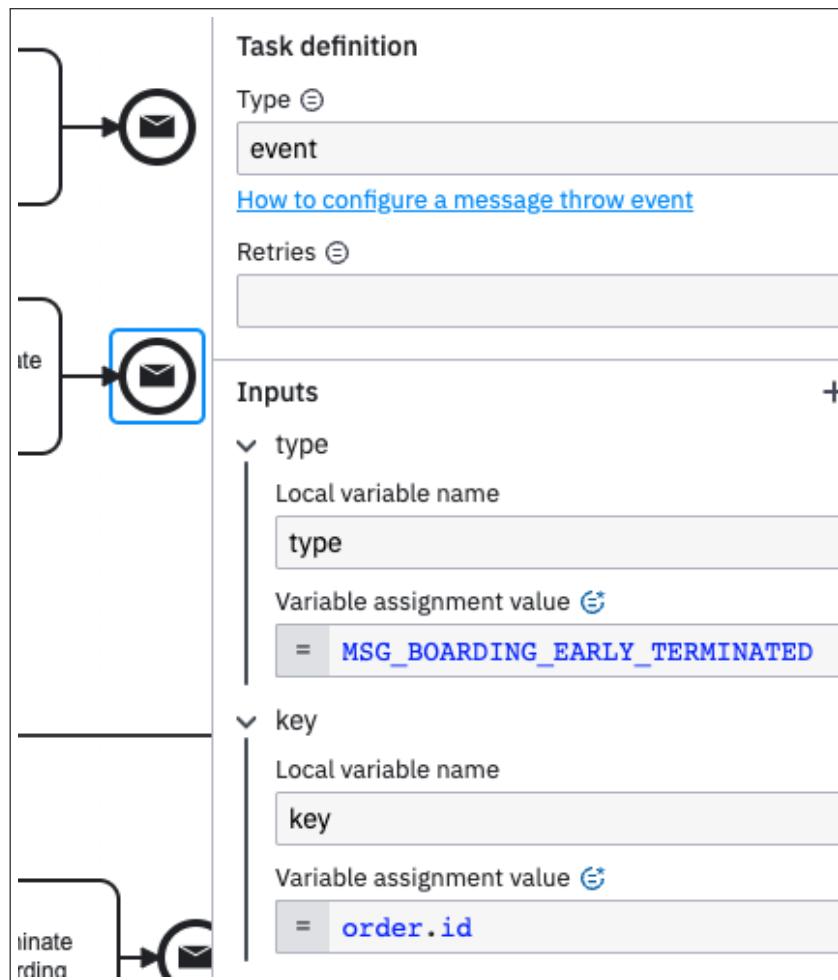


Obr. 5.54: Implementation: Call activity starting support process



Obr. 5.52: Implementation: Service task

Na konci podporného procesu je teda potrebné zasielať koncové udalosti. Na servery tak bola vytvorená služba, ktorá všetky potrebné vstupy získava z *Message End Event* elementu (obrázok č.5.55) a vytvorí z nich udalosť (výpis kódu č.5.6).



Obr. 5.55: Implementation: Messages from support process

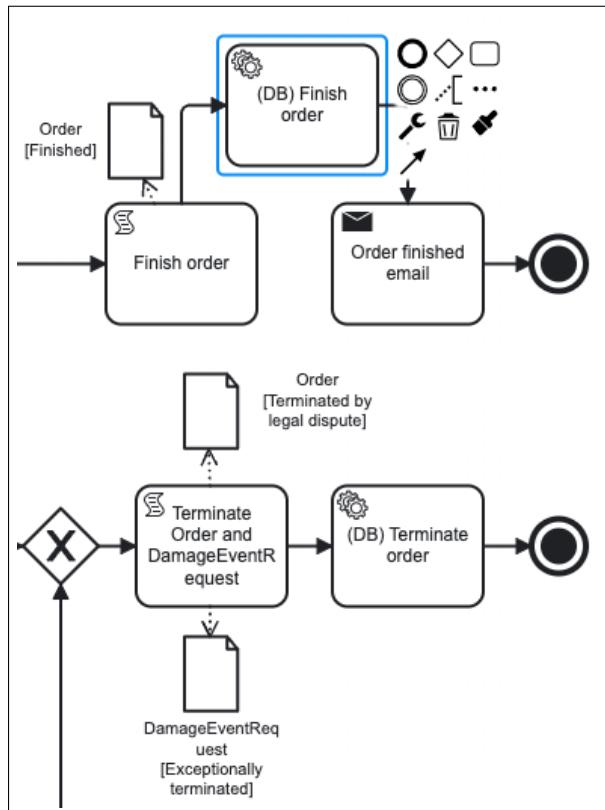
```

1 /**
2  * @type 'event'
3  * @input type -- message name
4  * @input key -- correlation key
5 */
6 zbc.createWorker({
7   taskType: 'event',
8   taskHandler: (job) => {
9     zbc.publishMessage({
10       name: job.variables.type,
11       correlationKey: job.variables.key
12     })
13     job.complete();
14   }
15 });

```

Výpis 5.6: Event worker

Na ďalších pracovných jednotkách si ukážeme ako sme vyriešili **ukladanie stavu objektov**. Pre potreby tejto práce sme si vystačili s ukladaním až na konci procesu pomocou služby (obrázok č.5.56). V skutočnosti by sme používali relačnú databázu ale pre zjednodušenie práce a vzhľadom na iné ciele diplomovej práce sme ukladali do textového súboru objekty, ktoré boli nevyhnutné na prechod celým procesom vo formáte JSON. Vo výpise kódu č.5.7 vidíme implementáciu ukladania do súboru.



Obr. 5.56: Implementation: Saving state

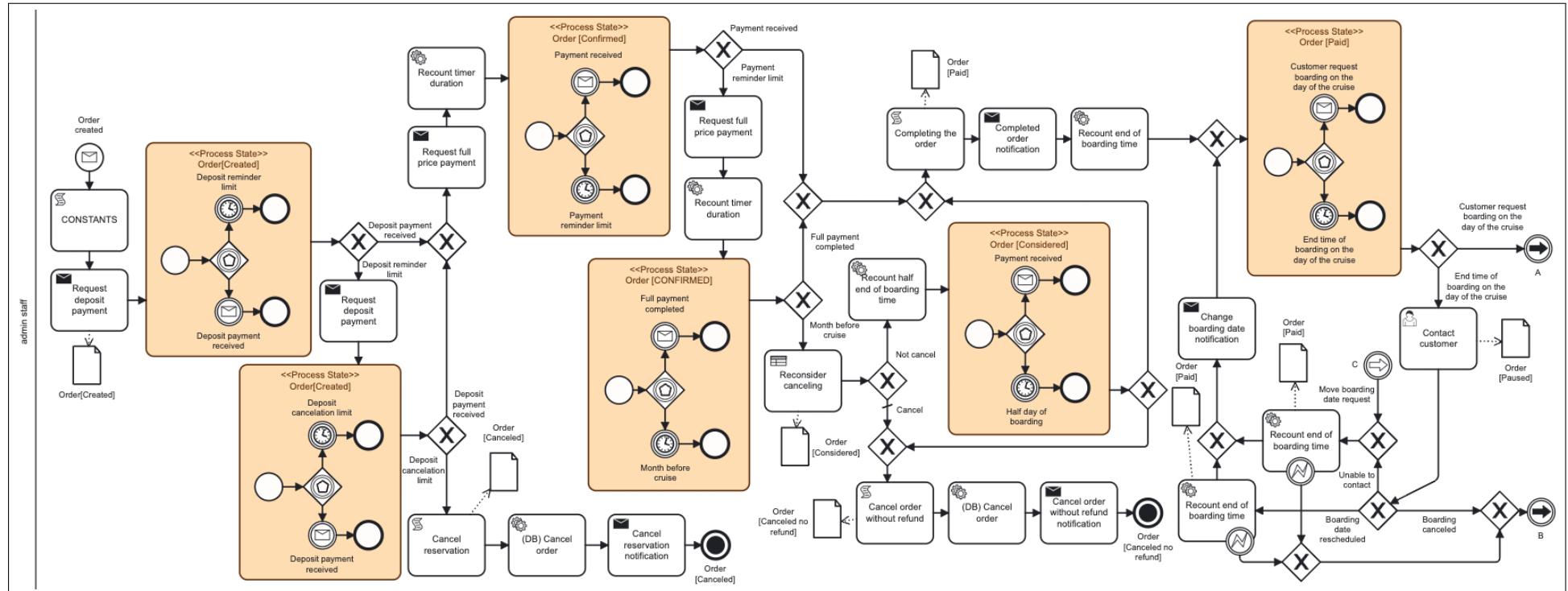
```

1 const fs = require('fs');
2 const PATH = 'db/orders.txt';
3 const saveOrderToDB = (order) => {
4     const data = fs.readFileSync(PATH);
5     const orders = JSON.parse(data);
6     const newState = orders.filter((item) => {
7         return item.order.id != order.id;
8     });
9     newState.push({order: order});
10    fs.writeFileSync(PATH, JSON.stringify(newState));
11 }

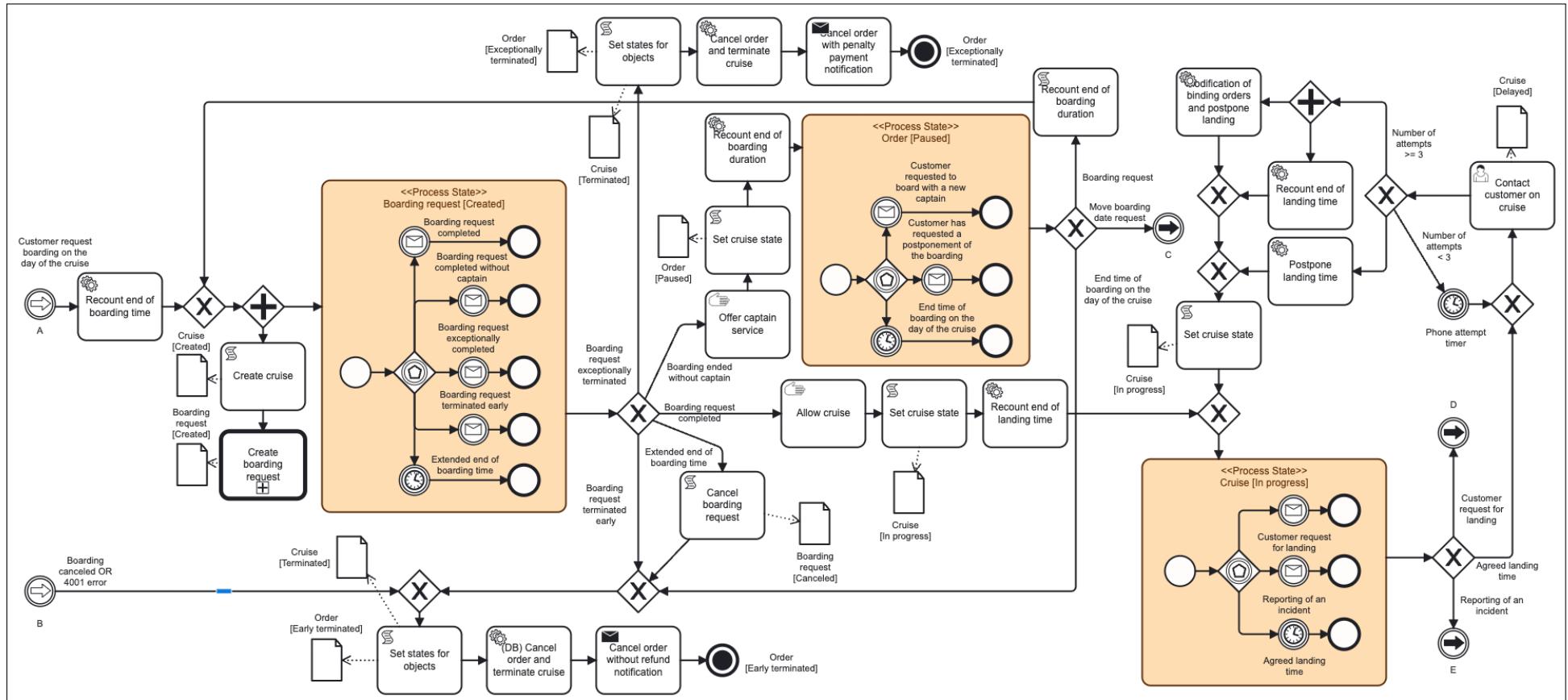
```

Výpis 5.7: Saving files

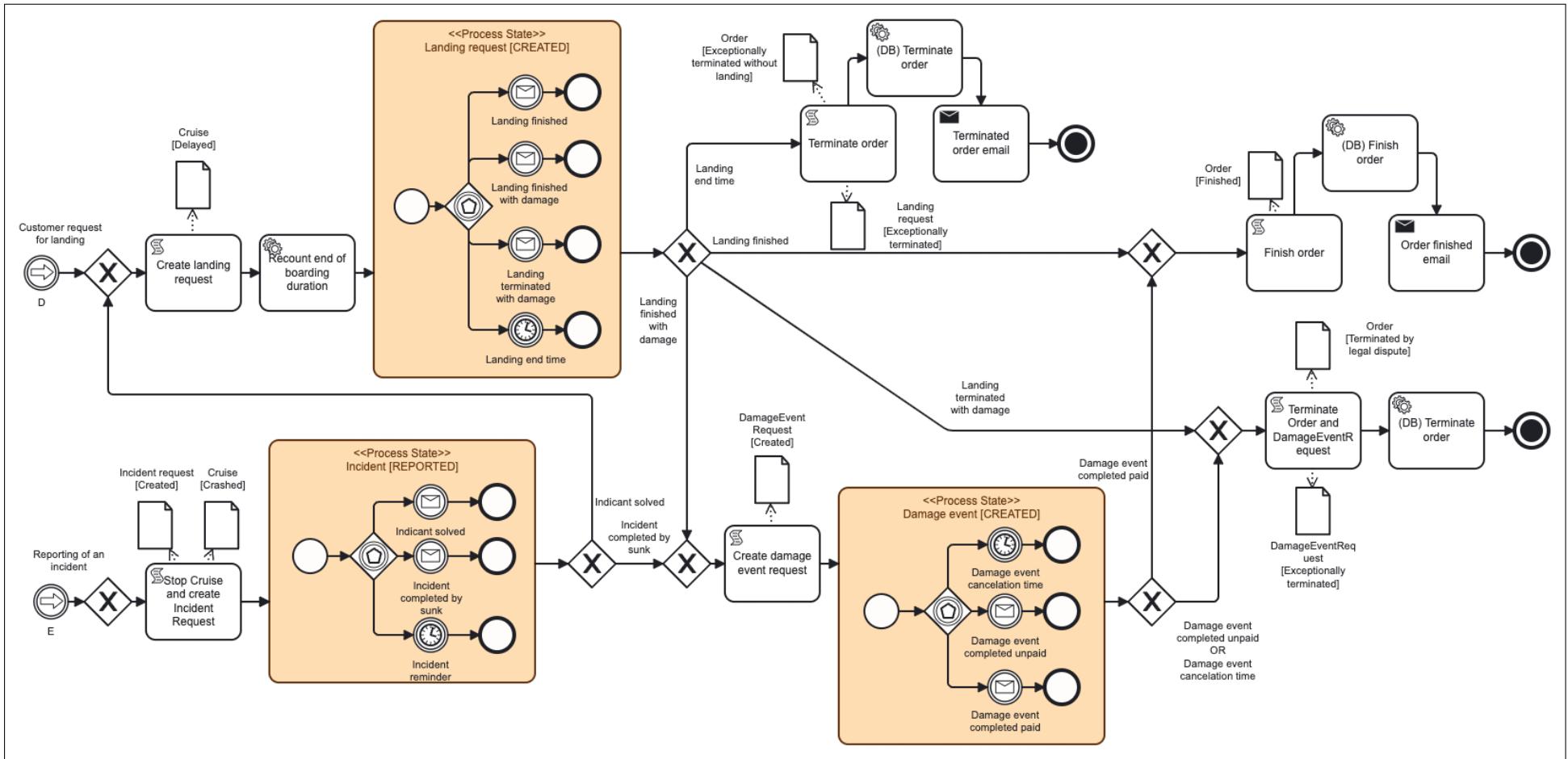
Postupným modelovaním pracovných jednotiek a ich konfiguráciu nám vznikli dva procesné implementačné diagramy. Na obrázkoch č.5.57, 5.58 a 5.59 vidíme implementačný model kľúčového diagramu. Na obrázku č.5.60 vidíme implementačný model podporného procesu *Boarding*.



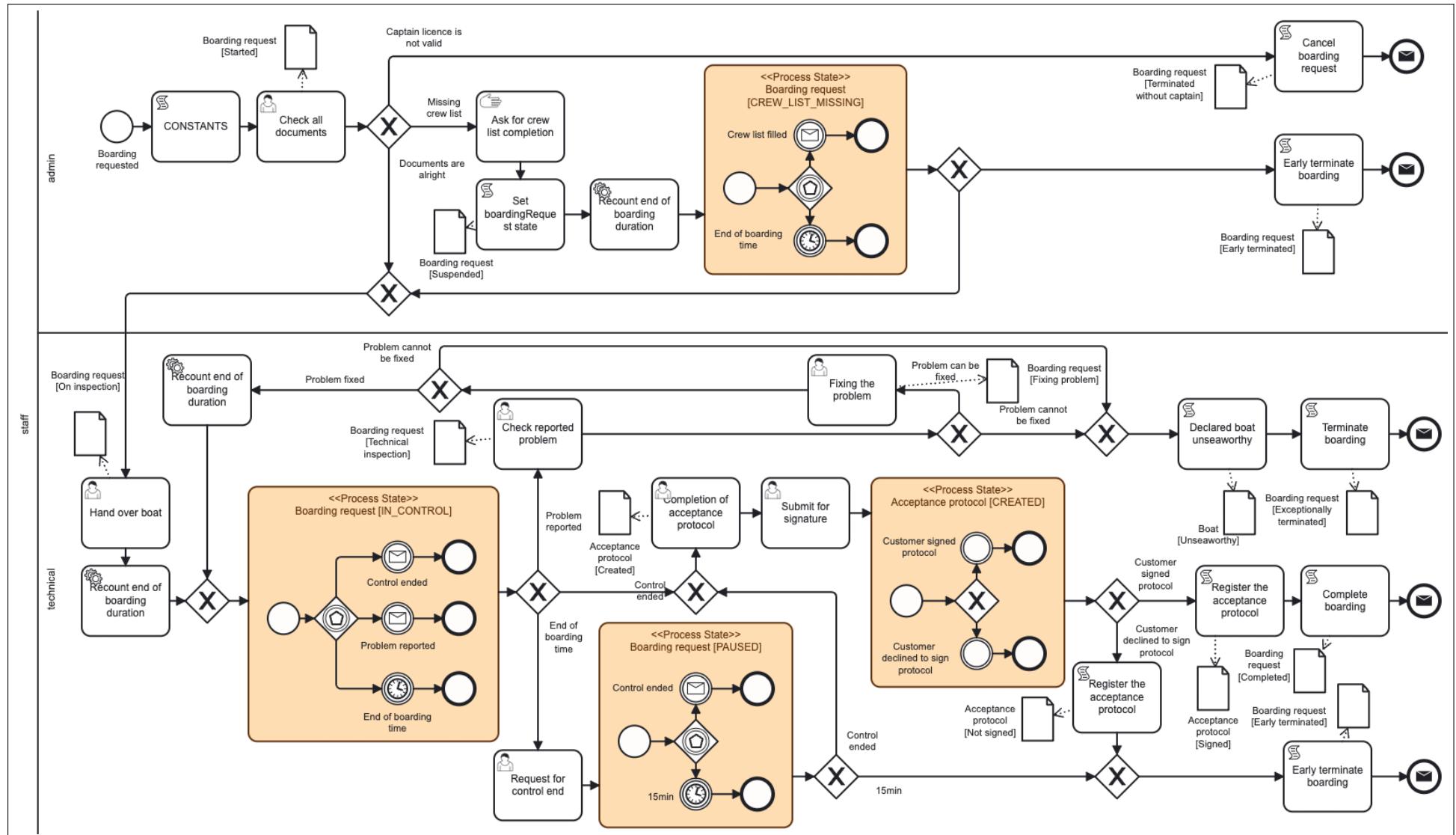
Obr. 5.57: Implementation model: process diagram of key process part 1



Obr. 5.58: Implementation model: process diagram of key process part 2



Obr. 5.59: Implementation model: process diagram of key process part 3



Obr. 5.60: Implementation model: process diagram of support process Boarding

### 5.2.5 Tvorba formulárov

Kľúčový ale hlavne podporný proces obsahuje aj úlohy priradzované ľuďom. Preto sme postupne podľa metodického postupu vytvárali formuláre. Vzniklo 6 formulárov, ktoré vidíme na obrázkoch 5.61 až 5.67.

Contact customer  
YachtCharter [KEY PROCESS]

Assigned      Unassign

Order n.: 1686999748781

Customer name: Milan Tester

Phone number: 802224011

Contact result\*

Unable to contact

Rescheduled boarding

Canceled boarding

Maximum is 3 days after 2024-06-18!

New boarding date\*

06/18/2024

Complete Task

Obr. 5.61: Implementation: Form for task Contact customer

Check all documents  
Boarding

Assigned      Unassign

Customer

Order n.: 1686999912559

Customer name: Milan Tester

Phone number: 696887589

Documents controll\*

Captain licence not valid

Captain licence valid, missing crew list

All documents are correct

Crew list number\*

Complete Task

Obr. 5.62: Implementation: Form for task Check all documents

Filter options	Hand over boat Boarding	Unassigned	Assign to me
All open			
<b>Hand over boat</b> Boarding Unassigned Created 17 Jun 2023 - 01:06 PM	<b>Customer</b> Order n.: 1686999912559 Customer name: Milan Tester Phone number: 696887589	<b>Boat</b> type: bavaria-38 name: Neptun 51 capacity: 6 dock: zkgj	
Contact customer YachtCharter [KEY PROCESS] Assigned to me Created 17 Jun 2023 - 01:02 PM			
Contact customer on cruise YachtCharter [KEY PROCESS] Assigned to me Created 17 Jun 2023 - 12:30 PM			
Check all documents			Complete Task

Obr. 5.63: Implementation: Form for task Hand over boat

Check reported problem Boarding	Unassigned
<b>Reported problem with boat</b>  Jib sheet needs to be replaced.  Fixable <input type="radio"/> Yes <input checked="" type="radio"/> No	
<b>Boat</b> type: bavaria-38 name: Neptun 51 capacity: 6 dock: zkgj	<b>Customer</b> Order n.: 1686999912559 Customer name: Milan Tester Phone number: 696887589
	Complete Task

Obr. 5.64: Implementation: Form for task Check reported problem

<p>Set state of problem Boarding</p> <h2>Reported problem with boat</h2> <p>Jib sheet needs to be replaced.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; padding: 5px;"><b>Boat</b></td> <td style="width: 50%; padding: 5px;"><b>Customer</b></td> </tr> <tr> <td style="padding: 5px;">type: bavaria-38</td> <td style="padding: 5px;">Order n.: 1686999912559</td> </tr> <tr> <td style="padding: 5px;">name: Neptun 21</td> <td style="padding: 5px;">Customer name: Milan Tester</td> </tr> <tr> <td style="padding: 5px;">capacity: 6</td> <td style="padding: 5px;">Phone number: 275500381</td> </tr> <tr> <td style="padding: 5px;">dock: zkgj</td> <td></td> </tr> </table> <h2>Solution</h2> <p>Problem fixed</p> <p><input checked="" type="radio"/> Yes  <input type="radio"/> No</p> <p>Describe how you fix the problem*</p> <p>I checked jib, foot part was damaged, take new one from storage and replace it</p> <p style="text-align: right; background-color: #e0e0e0; padding: 5px;"><b>Complete Task</b></p>	<b>Boat</b>	<b>Customer</b>	type: bavaria-38	Order n.: 1686999912559	name: Neptun 21	Customer name: Milan Tester	capacity: 6	Phone number: 275500381	dock: zkgj	
<b>Boat</b>	<b>Customer</b>									
type: bavaria-38	Order n.: 1686999912559									
name: Neptun 21	Customer name: Milan Tester									
capacity: 6	Phone number: 275500381									
dock: zkgj										

Obr. 5.65: Implementation: Form for task Fixing the problem

<p>Filter options All open</p> <ul style="list-style-type: none"> <li><b>Request for control end</b> Boarding Unassigned Created 17 Jun 2023 - 01:40 PM</li> <li>Contact customer on cruise YachtCharter [KEY PROCESS] Unassigned Created 17 Jun 2023 - 01:38 PM</li> <li>Contact customer on cruise YachtCharter [KEY PROCESS] Assigned to me Created 17 Jun 2023 - 01:21 PM</li> <li>Check all documents Boarding Unassigned</li> </ul>	<p>Request for control end Boarding</p> <p>Unassigned <b>Assign to me</b></p> <p><b>Customer info</b> Order n.: 1687001867074 Customer name: Milan Tester</p> <p><b>Boat info</b> Boat: Neptun 31 Dock number: 31y1</p> <p style="text-align: right; background-color: #e0e0e0; padding: 5px;"><b>Complete Task</b></p>
---	---

Obr. 5.66: Implementation: Form for task Request for control end

Contact customer on cruise  
YachtCharter [KEY PROCESS]

Assigned [Unassign](#)

OrderID: 1687000900252

Customer name: Milan Tester

Phone number: 989776862

- Cannot contact customer
- Customer cannot land today
- Move landing hour

Move todays landing time

End hours\*

18

- | +

End minutes\*

30

- | +

[Complete Task](#)

Obr. 5.67: Implementation: Form for task Contact customer on cruise

The screenshot shows a user interface for a task submission. At the top left, there is a 'Filter options' dropdown set to 'All open'. To the right, there are buttons for 'Assigned' and 'Unassign'. Below this, a section titled 'Customer' displays the name 'fcuuy 79pv9a6'. To the right, a section titled 'Boat' displays details: type 'bavaria-38', name 'Neptun 21', capacity '3', and dock 'ntgw'. A 'Protocol number\*' field contains the value '200536'. A 'Reported problems' field contains the text 'small scratch at bow on portside'. A checkbox labeled 'Customer signature' is checked. At the bottom right, a blue button says 'Complete Task'.

Obr. 5.68: Implementation: Form for task Submit for signature

### 5.2.6 Tvorba rozhodovacích diagramov a tabuliek

Klúčový proces obsahuje jedno business rozhodnutie. Bol vytvorený DRD diagram popisujúci vstupy ovplyvňujúce business rozhodnutie. Väčšina parametrov rozhodnutia získavame priamo ako vstupy z *Business Rule Task* úlohy. Jeden z parametrov (*Season*) bolo potrebné vypočítať pomocou krátkeho FEEL výraz. Ten podľa dátumov vyhodnocuje, či je objednávka urobená na sezónu alebo mimo sezónu. Na obrázku č.5.69 vidíme mapovanie vstupov, DRD diagram a výpočet parametru *Season*. Na obrázku č.5.70 vidíme rozhodovaciu tabuľku pre *Cancel decision* rozhodnutie.

### 5.2.7 Nasadenie diagramov, spustenie a vyladenie

Nasadzovali, spúšťali a ladili sme každú pracovnú jednotku. Na obrázkoch č.5.71 a č.5.72 vidíme priebeh procesu pri testovaní a ladení väčšieho množstva pracovných jednotiek pomocou rozhrania Operate.

The image shows the implementation of a business rule task. On the left, a BPMN diagram illustrates a process flow: a decision diamond "Full payment completed" leads to a gateway "X". From the gateway, one path goes to a task "Reconsider canceling" (with a local variable "cancelDecision"), which then leads to an "Order [Considered]" event. Another path from the gateway leads to a decision diamond "Month before cruise". From this diamond, one path leads to a task "Reconsider canceling" (with a local variable "cancelDecision"), which then leads to an "Order [Canceled no refund]" event. The other path from the diamond leads to a gateway "X".

**Called decision:**

- Decision ID:** cancelDecision
- Result variable:** cancelDecision

**Inputs:**

- seasonStart:**
  - Local variable name:** seasonStart
  - Variable assignment value:** `= date(string(now().year) + "-05-01")`
- seasonEnd:**
  - Local variable name:** seasonEnd
  - Variable assignment value:** `= date(string(now().year) + "-09-31")`

**Variables:**

- orderDate:**
  - Local variable name:** orderDate
  - Variable assignment value:** `= date(start_date)`
- boatCapacity:**
  - Local variable name:** boatCapacity
  - Variable assignment value:** `= number(boat.capacity)`
- customerCredibility:**
  - Local variable name:** customerCredibility
  - Variable assignment value:** `= number(customer.credibility)`

**DRD Diagram:**

```

graph TD
    Cancel[Cancel decision] --> Season[Season]
    Season --> orderDate((orderDate))
    Season --> seasonStart((seasonStart))
    Season --> seasonEnd((seasonEnd))
    boatCapacity((boatCapacity)) --> Season
    customerCredibility((customerCredibility)) --> Season
  
```

**Season:**

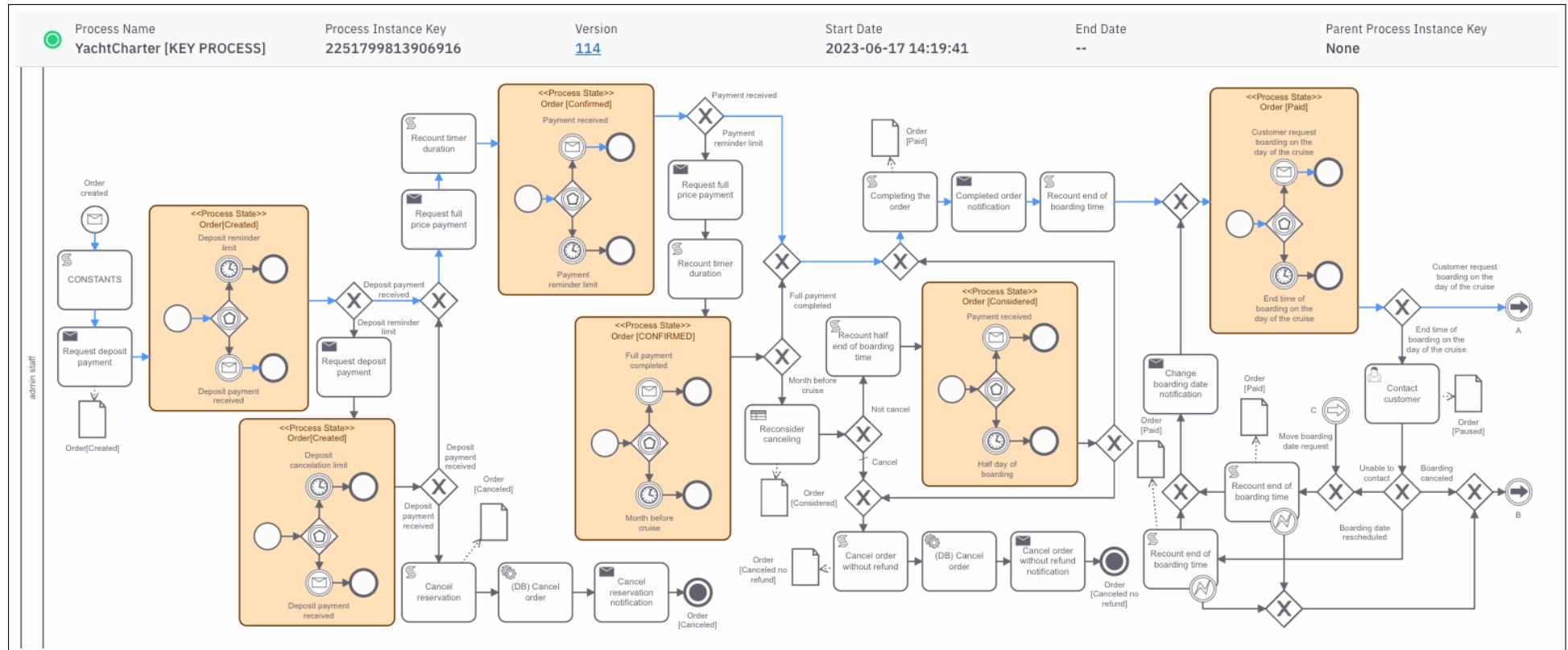
**Condition:** `orderDate >= seasonStart and orderDate <= seasonEnd`

Obr. 5.69: Implementation: Inputs mapping and DRD diagram

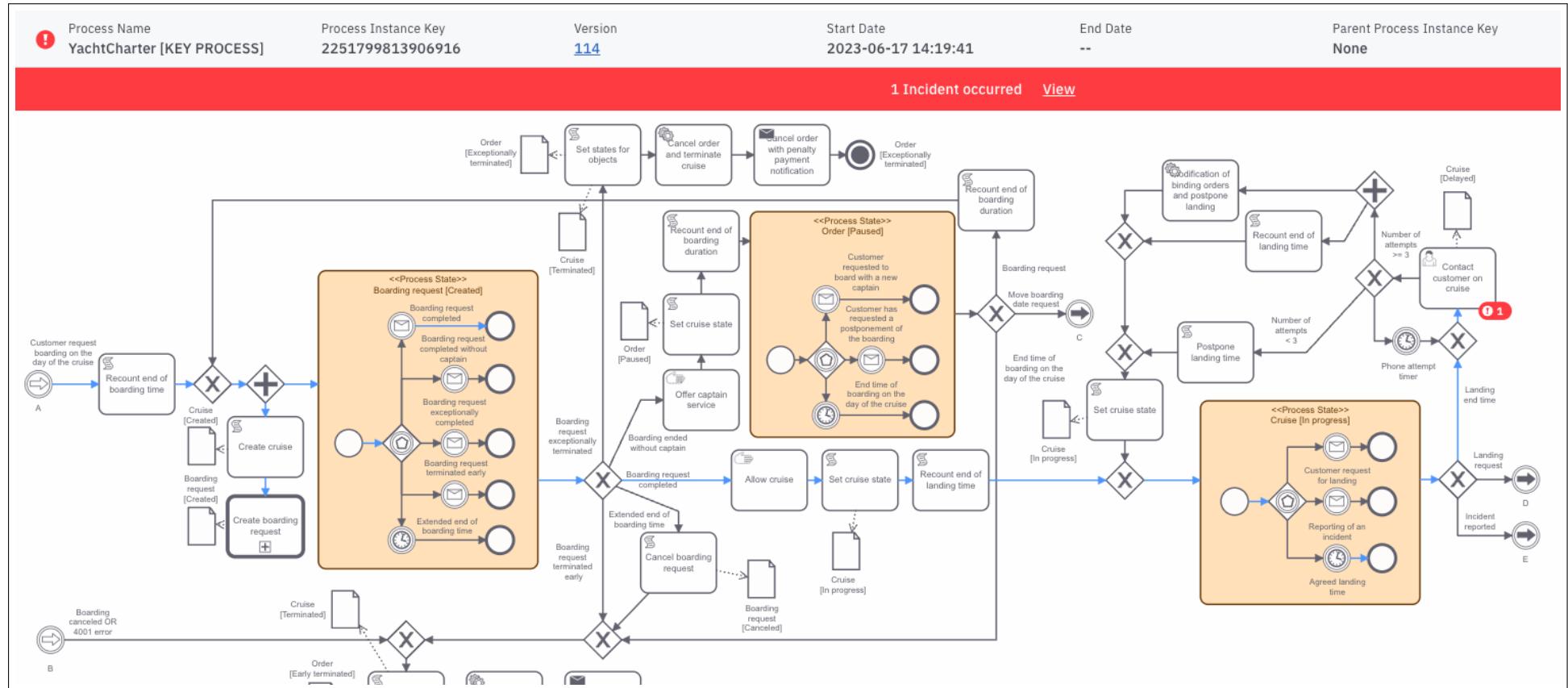
Decision 1 | Hit Policy: First

	When season boolean	And boatCapacity number	And customerCredibility number	Then cancelDecision boolean	Annotations
1	true	[3..6]	>= 85	false	Good credibility even for popular boats during season.
2	true	[3..6]	< 85	true	Not enough credibility even for popular big boats during season.
3	true	-	>= 50	false	Enough credibility for less charterable boats in season.
4	true	-	< 50	true	
5	false	<= 3	>= 50	true	Small boats are popular off-season.
6	false	> 3	-	false	Big boats are not so popular off-season.

Obr. 5.70: Implementation: Decision table



Obr. 5.71: Implementation: Testing a debugging in Operate 1



Obr. 5.72: Implementation: Testing a debugging in Operate 2

### **5.3 Zhodnotenie nástroja Camunda 8**

Pri vytváraní tejto kapitoly sme prirodzene získali isté skúsenosti s workflow nástrojom Camunda 8. V práci sme používali Self-managed verziu. Táto verzia si vyžaduje isté všeobecnejšie vývojárske zručnosti a neponúka všetky možnosti, ktoré má SaaS verzia. My sme ale mali možnosť vyskúšať si SaaS verziu len s obmedzenou funkcionalitou, preto metodický postup nemusí zahŕňať úplne všetky možné kroky. Napríklad sme nemali možnosť vytvárať užívateľské role, a tento krok tak nie je súčasťou metodického postupu.

Platforma je jej autormi označovaná ako orchestračný nástroj mikroservíš na manažovanie podnikových procesov. To znamená, že nemusí neponúkať všetky možnosti, na ktoré môžeme byť zvyknutí pri iných platformách (napr. vytváranie udalostí z užívateľského rozhrania nástroja). Pre daný účel ale ponúka veľmi dobre vlastnosti ako horizontálna škálovateľnosť, vysoká dostupnosť a odolnosť voči chybám.

Pri modelovaní sa pracuje s rozhraniami, ktoré sú používané aj v iných nástrojoch (projekt BPMN.iO) a sú tak veľmi dobre vyladené a práca s nimi je veľmi intuitívna. Zo začiatku sme boli trochu sklamaný, pretože naprieč pestrej palete formulárového rozhrania nástroj reálne nepodporoval funkcionality ako vytváranie stĺpcov pre vstupy alebo vstup typu dátum. Neskôr ale vyšla nová verzia (z 8.1 na 8.2), ktorá tieto funkcionality doplnila. Zároveň s novou verzou došlo k výraznému vyladeniu docker images, pomocou ktorých Self-managed verzia funguje, a prestalo tak dochádzať k občasnému výpadkom alebo dlhým načítavaniam rozhraní Tasklist a Operate.

Verzia 8 je oproti pôvodnej verzii 7 stále dosť mladá. Je ale postavená na novej architektúre, ktorá prináša vyššie spomenuté vlastnosti a je vidieť, že jej vývoj stále napreduje. Určite by sme workflow nástroj Camunda 8 doporučili ako vhodný nástroj na automatizáciu podnikových procesov.

# 6. Zhrnutie metodického postupu a jeho diskusia

V tejto kapitole zhrnieme celý metodický postup transformácie business procesov do aplikáčnych procesov nástroja Camunda. Ten sa skladá z aplikovania dvoch metodických postupov na odlišné typy modelov. Následne budeme diskutovať zmeny a prínosy metodického postupu.

## 6.1 Transformácia konceptuálnych modelov na technologické modely

1. **Špecifikácia použitej technológie so zameraním na možnú zmenu granularity vykonávaných úloh** [plné znenie v (Tauchmanová 2023, str. 114-115)]
  - 1.1. analýza spôsobov vykonania procesu s prihliadnutím na využívanie rôznych technológií
  - 1.2. špecifikácia konkrétnego workflow v závislosti na konkrétnom technologickom spôsobe
  - 1.3. vytvorenie procesného diagramu pre daný proces so zohľadnením použitej technológie
2. **Špecifikácia technických rolí** [plné znenie v (Tauchmanová 2023, str. 115)]
  - 2.1. vyhľadať aktérov či technické role v modeli business systému a jeho dokumentácií
  - 2.2. vytvoriť zoznam technických rolí/aktérov
  - 2.3. vytvoriť plavebné dráhy pre konkrétné role v procesných diagramoch
3. **Špecifikácia externých udalostí** [plné znenie v (Tauchmanová 2023, str. 116)]
  - 3.1. identifikovať všetky udalosti v detailných procesných mapách
  - 3.2. každú udalosť zanalyzovať, či sa skutočne jedná o udalosť prichádzajúca z vonkajšieho prostredie IS a ako bude táto udalosť vykonaná (doplniť možnosť úlohy v systéme/ad hoc procesu)
  - 3.3. vytvoriť zoznam udalostí s požadovanými informáciami
  - 3.4. pridať vhodným spôsobom externé udalosti do procesných diagramov do príslušných plavebných dráh

- 4. Transformácia úloh a pridanie potrebných úloh v závislosti na alternatívnych prechodoch** [plné znenie v (Tauchmanová 2023, str. 116-117)]
  - 4.1. preskúmať všetky detailné procesné mapy a identifikovať typ všetkým úloham,
  - 4.2. doplniť úlohy s korešpondujúcimi typmi do príslušných plavebných dráh,
  - 4.3. doplniť novo identifikované úlohy s korešpondujúcim typom do príslušných plavebných dráh,
  - 4.4. doplniť udalosti pre štart/y a konce procesu,
  - 4.5. doplniť spojenie vrátane príslušných brán medzi jednotlivými úlohami a udalosťami,
  - 4.6. doplniť podmienky/popisy u XOR brán.
- 5. Určenie vstupov a výstupov pre každú úlohu procesu** [plné znenie v (Tauchmanová 2023, str. 118)]
  - 5.1. pre každý procesný diagram pripraviť tabuľku
  - 5.2. do tabuľky vyplniť názov úlohy a identifikovať a zapísat potrebné vstupy a výstupy
  - 5.3. prípadne doplniť zodpovedné roly danej úlohy
  - 5.4. nejasnosti si vyjasniť s vlastníkom procesu
- 6. Špecifikácia business premenných** [plné znenie v sekcií č.4.3.2]
  - 6.1. preskúmanie časových udalostí v procesnom diagrame,
  - 6.2. vytvorenie zoznamu business premenných,
  - 6.3. vyhľadať business premenné v zozname vstupov a výstupov úloh,
  - 6.4. diskutovať zoznam business premenných s vlastníkom procesu,
  - 6.5. nahradiť hodnoty časových udalostí názvom premennej.
- 7. Aktualizácia diagramu tried** [plné znenie v (Tauchmanová 2023, str. 118–119)]
  - 7.1. prejsť všetky procesné diagrame a skontrolovať, prípadne doplniť, chýbajúce triedy alebo atribúty vrátane dátových typov,
  - 7.2. do diagramu tried doplniť stavové premenné,
  - 7.3. skontrolovať, prípadne pridať, technické role/aktéri v diagrame tried
  - 7.4. skontrolovať, prípadne doplniť, dátové typy, povinnosti dátových typov a väzieb medzi jednotlivými triedami
- 8. Špecifikácia notifikácií** [plné znenie v (Tauchmanová 2023, str. 119)]
  - 8.1. identifikovať úlohy, v ktorých sú notifikácie poslané,
  - 8.2. diskutovať znenie textov notifikácií, vrátane odosielateľov a príjemcov, s vlastníkom procesov, prípadne s kľúčovými užívateľmi,
  - 8.3. vytvoriť tabuľku pre každú notifikáciu, ktorá obsahuje potrebné informácie na základe charakteru notifikácie,
  - 8.4. identifikovať výskyt business premenných v notifikáciách a korektne ich upraviť v tabuľke notifikácií, [sekcia č.4.3.3]
  - 8.5. identifikovať výskyt špeciálnych premenných v notifikáciách a korektne ich upraviť v tabuľke notifikácií. [sekcia č.4.3.3]

9. **Pridanie rozhodovacích modelov a tabuliek** [plné znenie v sekcii č.4.3.1]
  - 9.1. identifikovať vstupy, výstupy a zdroj znalostí pre business rozhodnutie,
  - 9.2. vytvoriť DRD pre všetky identifikované business rozhodnutia,
  - 9.3. vytvoriť prvý návrh rozhodovacích tabuliek pre každé rozhodnutie z DRD diagramov,
  - 9.4. diskutovať štruktúru business rozhodnutí s vlastníkom procesu,
  - 9.5. finalizovať rozhodovacie tabuľky.
10. **Pridanie ďalšieho popisu či poznámok**

## 6.2 Transformácia technologických modelov na implementačné modely

Plné znenie nájdeme v sekcii č..

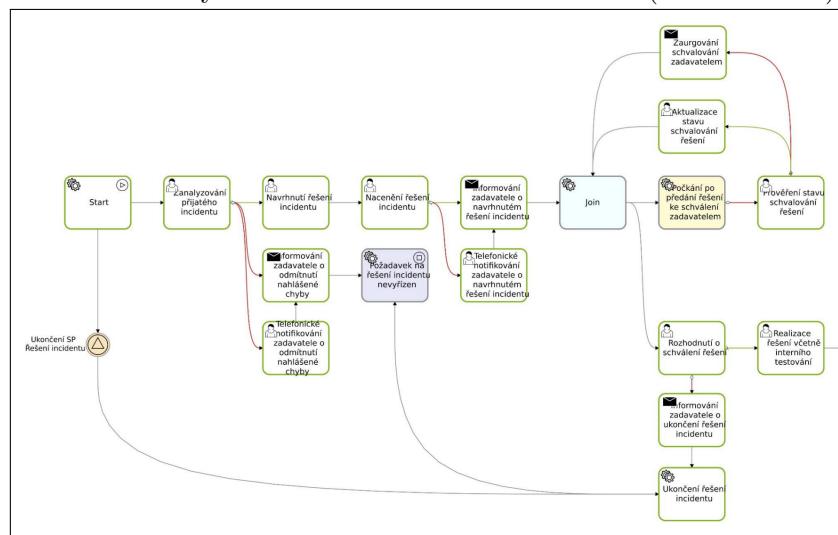
1. **Vytvorenie štruktúry projektu** [sekcia č.5.1.1]
2. **Výber pracovnej jednotky** [sekcia č.5.1.2]
3. **Vytvorenie procesného diagramu** [sekcia č.5.1.3]
  - 3.1. vytvorenie BPMN diagramu
  - 3.2. pridanie plavebných dráh
  - 3.3. konfigurácia vstupnej udalosti
  - 3.4. pridanie zoznamu premenných a konštánt
4. **Transformácia procesu** [sekcia č.5.1.4]
  - 4.1. modelovanie procesného stavu
  - 4.2. konfigurácia výstupu zo stavu a stavových udalostí
  - 4.3. modelovanie úloh a brán
  - 4.4. konfigurácia úloh a brán
  - 4.5. modelovanie objektov
  - 4.6. konfigurácia objektov a zmeny ich stavu
5. **Tvorba formulárov** [sekcia č.5.1.5]
  - 5.1. vytvorenie súboru formulára
  - 5.2. modelovanie vstupov
  - 5.3. konfigurácia vstupov
  - 5.4. napojenie na proces
  - 5.5. mapovanie formulárových vstupov na výstupy úlohy
6. **Tvorba rozhodovacích diagramov a tabuliek** [sekcia č.5.1.6]
  - 6.1. vytvorenie súboru pre rozhodovací diagram
  - 6.2. transformácia DRD diagramu
  - 6.3. naplnenie rozhodovacej tabuľky
7. **Nasadenie diagramov** [sekcia č.5.1.7]
8. **Spustenie diagramov a ich vyladenie** [sekcia č.5.1.8]

### 6.3 Diskusia

Problematike transformácie business procesov do aplikačných procesov sa už venoval Pavelčák (2021). Vytvoril metodický postup transformácie do technologického modelu. Ten overila a mierne upravila Tauchmanová (2023). My sme v metodickom postupe našli drobné nedostatky. Postup nereflektuje automatizované business rozhodnutia. Tie môžeme vo vybraných workflow nástrojoch implementovať pomocou DMN notácie. Preto sme navrhli rozšírenie metodického postupu o tvorbu rozhodovacích diagramov a tabuľiek.

Ďalším nedostatkom, ktorý sa nám podarilo identifikovať je znížená udržateľnosť a modifikovateľnosť celkovej architektúry. Je možné, že tieto vlastnosti by boli posilnené kvalitnou implementáciou ale nie je vhodné sa nato spoliehať. Preto sme navrhli rozšírenie metodického postupu o zoznam business premenných, ktoré majú potenciál sa časom alebo nejakým business rozhodnutím meniť. Tento zoznam má pri implementácii slúžiť ako zoznam konfigurovatelných premenných. Práca teda mierne rozšírila a upravila metodický postup transformácie konceptuálnych modelov do technologických modelov.

Pozrime sa na rozdieli medzi metodickým postupom pre transformáciu technologických modelov do implementačných modelov od autorov Pavelčák a Tauchmanová a našim metodickým postupom. Hlavným rozdiel je odlišný implementačný nástroj. Oni si zvolili workflow nástroj TeamAssistant (TAS). My sme v práci zvolili nástroj Camunda. Prínosom práce je tak prvý ustálený metodický postup transformácie technologických modelov do implementačných modelov pre nástroj Camunda. Z hľadiska vývojára ma Camunda oproti TeamAssistan výhodu, že podporuje štandard BPMN 2.0. Rozdiel medzi technologickým a implementačným modelom potom nie je až taký veľký. TAS sice používa na modelovanie rovnaké rozhranie ale modely sa nedržia BPMN 2.0 štandardu (obrázok č.6.1).



Obr. 6.1: TeamAssistant model (Tauchmanová 2023, str. 100)

Na druhú stranu TAS poskytuje funkcionality ako manuálne vytváranie udalostí z jeho rozhrania, či pred-implementované odosielanie emailov. Camunda ale poskytuje skvelé Operate rozhrania na ladenie procesov. Camunda má novšiu modernejšiu architektúru

postavenú pre dosahovanie dobrej horizontálnej škálovateľnosti a vysokej dostupnosti. TAS ponúka veľké množstvo predprogramovaných funkcionalít. Bez hlbšej analýzy a parametrov, ktoré podnik požadu sa nedá úplne jednoznačne povedať, ktorý nástroj je vhodnejší pre automatizáciu podnikových procesov. Pri výbere bude hrať veľkú rolu veľkosť procesov, požadovaný výkon a miera vlastnej implementácie.

Drobným rozdielom medzi metodickými postupmi je aj prístup ku implementácií. Pavelčák a Tauchmanová v postupe nijako nedoporučuje rozdelenie implementácie do menších celkov. Toto je dosť rizikové pri väčších procesoch a môže to spomalíť a znepríjemniť vývoj a ladanie.

Kombináciu dvoch metodických postupov, ktoré sme zhrnuli v tejto kapitole, umožňuje čitateľ na základe konceptuálnych modelov business procesov vytvoriť spustiteľnú procesné riadenú aplikáciu v modernej a stále rozvíjajúcej sa platforme Camunda.



# Záver

Výsledok diplomovej práce je overený metodický postup transformácie konceptuálnych modelov do technologických modelov, ktorý bol následne doplnený a mierne upravený.

Metodický postup od Pavelčáka, ktorý zrevidovala Tauchmanová bol súčasťou kompletný pre veľké množstvo podnikových procesov. Nezohľadňoval ale tie procesy, ktorých súčasťou sú komplexné rozhodovacie pravidlá. Tie môžu byť modelované pomocou DMN notácie a automatizované pomocou rozhodovacích tabuliek. Metodický postup bol preto doplnený o nepovinný krok, ktorý pokryje procesy obsahujúce automatizované business rozhodovanie.

Metodického postupu obsahoval ďalší drobný nedostatok, ktorý nie je na prvý pohľad jednoduché objaviť. Išlo o zníženú udržateľnosť a modifikovateľnosť architektúry, ktorá sa prejavuje až časom pri zmenách v procese alebo informačnom systéme. Preto bol metodický postup doplnený o vytváranie zoznamu business premenných, ktoré majú tendenciu meniť svoje hodnoty časom alebo business rozhodnutím. Na prvý pohľad môže tento krok pôsobiť zbytočne a komplikovať. V praxi sa ale často stáva, že veci ktoré mali byť stále a nemenné sa stávajú nestálymi a menia sa. Vo väčšine týchto prípadov je potom finančne náročnejšia oprava takýchto nedostatkov ako väčší čas strávený pri analýze a návrhu.

Ďalším výsledkom je nový metodický postup transformácie technologických modelov do implementačných modelov nástroja Camunda, ktorý bol následne aplikovaný na konkrétnom business procese.

Diplomová práca tak splnila všetky ciele, ktoré boli v jej úvode stanovené.

## Prínosy a rozšírenia

Prínosom práce je metodický postup, ktorý môže byť použitý pri transformácii podnikových procesov do workflow nástroja Camunda alebo ako zdroj pre porovnanie implementačnej náročnosti rôznych workflow nástrojov. Prínosom je aj ukážka aplikovania metodiky MMABP na komplexnejšom business procese z netradičného podnikateľského oboru, ktorý v poslednej dobe nabera veľké množstvo nových zákazníkov. Možným akademickým prínosom je použitie postupu ako pomôcku vo výuke študentov business analýzy. Možným prínosom mimo akademickej pôdy je priblíženie podnikového riadenia a procesov programátorom.

Možným rozšírením práce s prínosom mimo akademickej pôdy je overenie postupu na reálnom komplexnom business procese. Aby mohla byť práca používana vo výuke business analytikov bolo by vhodné odstrániť nedostatky nástroja spôsobené jeho účelom, ktoré v praxi sice nemusia byť významné ale pre výuku sú značne nepríjemné. Doplnením by tak mohlo byť vytvorenie systému, ktorý umožní vytvárať, spravovať v procese používané udalosti.



# Zoznam použitej literatúry

- Weske, Mathias (2019). Business Process Management. 3rd ed.  
Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-662-59431-5.  
DOI: 10.1007/978-3-662-59432-2. (Cit. 25. 11. 2022).
- Sneed, Harry a Chris Verhoef (jún 2001).  
“Reengineering the Corporation - A Manifesto for IT Evolution”. In:  
URL: [https://www.researchgate.net/publication/2366189\\_Reengineering\\_the\\_Corporation\\_-\\_A\\_Manifesto\\_for\\_IT\\_Evolution](https://www.researchgate.net/publication/2366189_Reengineering_the_Corporation_-_A_Manifesto_for_IT_Evolution).
- Stiehl, Volker (jan. 2014). Process-driven applications with BPMN. en. 2014. vyd.  
Springer International Publishing. ISBN: 978-3-319-07218-0.  
DOI: 10.1007/978-3-319-07218-0.
- Dragoni, Nicola et al. (2017). “Microservices: Yesterday, Today, and Tomorrow”. In:  
Present and Ulterior Software Engineering. Ed. Manuel Mazzara a Bertrand Meyer.  
Cham: Springer International Publishing, s. 195–216. ISBN: 978-3-319-67425-4.  
DOI: 10.1007/978-3-319-67425-4\_12.  
URL: [https://doi.org/10.1007/978-3-319-67425-4\\_12](https://doi.org/10.1007/978-3-319-67425-4_12).
- Řepa, Václav (2021). Information Modelling of Organizations. Oeconomica.  
ISBN: 978-80-245-2441-2. URL: <https://oeconomica.vse.cz/publikace/information-modelling-of-organizations/>.
- Pavelčák, Milan (2021). “Transformace podnikových procesů na spustitelné procesy”.  
Diplomová práce. Praha: Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií.  
URL: [https://vskp.vse.cz/83078\\_transformace-podnikovych-procesu-na-spustitelne-procesy](https://vskp.vse.cz/83078_transformace-podnikovych-procesu-na-spustitelne-procesy).
- Tauchmanová, Michaela (2023).  
“Transformace business procesů do spustitelných aplikáčních procesů”. Diplomová práce. Praha: Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky.
- Hevner, Alan R. et al. (2004). “Design Science in Information Systems Research”.  
In: MIS Quarterly 28.1, s. 75–105. ISSN: 02767783.  
URL: <http://www.jstor.org/stable/25148625> (cit. 05. 12. 2022).
- Camunda Platform 8 Docs (2023). URL: <https://docs.camunda.io> (cit. 11. 05. 2023).
- Freund, Jakob a Bernd Rücker (2019). Real-Life BPMN. Includes an introduction to DMN.  
4th ed. Independently published. ISBN: 9781086302097.
- Sommerwille, Ian (2010). Software engineering. 9th ed. Pearson. ISBN: 978-0-13-703515-1.
- Benington, H. D. (1987). “Production of Large Computer Programs”.  
In: Proceedings of the 9th International Conference on Software Engineering. ICSE '87.  
Monterey, California, USA: IEEE Computer Society Press, s. 299–310.  
ISBN: 0897912160. DOI: 10.5555/41765.41799.  
URL: <https://dl.acm.org/doi/pdf/10.5555/41765.41799>.
- Royce, W. W. (1987).  
“Managing the Development of Large Software Systems: Concepts and Techniques”.

- In: Proceedings of the 9th International Conference on Software Engineering. ICSE '87.  
Monterey, California, USA: IEEE Computer Society Press, s. 328–338.  
ISBN: 0897912160. DOI: 10.5555/41765.41801.  
URL: <https://dl.acm.org/doi/pdf/10.5555/41765.41801>.
- Řepa, Václav (2007). Podnikové procesy. procesní řízení a modelování.  
2., aktualiz. a rozš. vyd. Praha: Grada. ISBN: 978-80-247-2252-8.
- Alexey, Kim (2017). “Analýza možnosti elektronizace podnikových procesů”.  
Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních  
technologií, katedra softwarového inženýrství.  
URL: <https://dspace.cvut.cz/handle/10467/68287>.
- Lipowski, Adam (2022). “Aplikace pro hodnocení zaměstnanců ČVUT FEL”.  
Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta  
elektrotechnická, Centrum znalostního managementu.  
URL: <https://dspace.cvut.cz/handle/10467/101697>.
- Řepa, Václav (2022). “Coping with natural parallelism in business process models”.  
In: BIR 2022 Workshops and Doctoral Consortium. Zv. 3223.  
Rostock, Germany: CEUR Workshop Proceedings, s. 24–34.  
URL: <https://ceur-ws.org/Vol-3223/paper3.pdf>.
- Hammer, Michael a James Champy (1993).  
“Reengineering the corporation: A manifesto for business revolution”.  
In: Business Horizons 36.5, s. 90–91. ISSN: 0007-6813.  
DOI: [https://doi.org/10.1016/S0007-6813\(05\)80064-3](https://doi.org/10.1016/S0007-6813(05)80064-3).  
URL: <https://www.sciencedirect.com/science/article/pii/S0007681305800643>.
- Martin, Robert C. (c2014). Agile software development, principles, patterns, and practices.  
First Edition. Harlow: Pearson. ISBN: 978-1-292-02594-0.
- Řepa, Václav (2012). Procesně řízená organizace. 1. vyd. Praha: Grada.  
ISBN: 978-80-247-4128-4.
- Eriksson, Hans-Erik a Magnus Penker (c2000).  
Business Modeling with UML. Business Patterns at Work. Wiley. ISBN: 0471295515.
- Group, The Open (2018). The TOGAF Standard - Version 9.2. 11th ed. Van Haren.  
ISBN: 978-9401802833.
- Řepa, Václav a Oleg Svatoš (2021).  
“Adaptive and Resilient Business Architecture for the Digital Age”. In:  
Architecting the Digital Transformation.  
Digital Business, Technology, Decision Support, Management.  
Ed. Alfred Zimmermann, Rainer Schmidt a Lakhmi C. Jain.  
Springer International Publishing, s. 199–221. ISBN: 978-3-030-49640-1.  
DOI: 10.1007/978-3-030-49640-1\_11.  
URL: [https://doi.org/10.1007/978-3-030-49640-1\\_11](https://doi.org/10.1007/978-3-030-49640-1_11).
- Group, Object Management (2011). Business Process Model and Notation (BPMN). 2.0.  
URL: <http://www.omg.org/spec/BPMN/2.0> (cit. 30.05.2023).
- (2017). OMG Unified Modeling Language (OMG UML). 2.5.1.  
URL: <https://www.omg.org/spec/UML/2.5.1> (cit. 30.05.2023).

Concepts (2023). What is Camunda Platform 8? Camunda Platform 8 Docs.

URL: <https://docs.camunda.io/docs/components/concepts/what-is-camunda-platform-8/> (cit. 12.05.2023).

DEEHAN, NIALL (2023).

Camunda Platform 8 for Camunda Platform 7 Users. What You Need to Know.

URL: <https://camunda.com/blog/2022/04/camunda-platform-8-for-camunda-platform-7-users-what-you-need-to-know/> (cit. 12.05.2023).

Overview Components (2023). Camunda Platform 8 Docs.

URL: <https://docs.camunda.io/docs/components/> (cit. 12.05.2023).

Working with APIs and tools (2023). Camunda Platform 8 Docs.

URL: <https://docs.camunda.io/docs/apis-tools/working-with-apis-tools/> (cit. 12.05.2023).

Identity (2023). Installation and first steps. Camunda Platform 8 Docs.

URL: <https://docs.camunda.io/docs/self-managed/identity/getting-started/install-identity/> (cit. 01.06.2023).

Camunda Platform 8 Pricing (2023).

URL: <https://camunda.com/pricing/> (cit. 11.05.2023).

Koziolek, Heiko (2011).

“Sustainability evaluation of software architectures. A systematic review.” In:

Proc. 7th Int. ACM/SIGSOFT Conf. on the Quality of Software Architectures (QoSA’11). ACM.

Decision Model and Notation (2023). 1.4. Object Management Group.

URL: <https://www.omg.org/spec/DMN/> (cit. 08.06.2023).

Timer events (2023). Camunda Platform 8 Docs.

URL: <https://docs.camunda.io/docs/components/modeler/bpmn/timer-events/> (cit. 13.06.2023).



## **Prílohy**



# A. Elektronická príloha

Repozitár projektu sa nachádza na adrese:

<https://github.com/marcel-zec/masters-thesis>

Popis spustenie projektu sa nachádza na adrese:

<https://github.com/marcel-zec/masters-thesis#readme>

## Prerekvizity

- nainštalovaný Docker 20.10.16+
  - odkaz na stiahnutie: <https://docs.docker.com/get-docker/>
- nainštalovaný NodeJS v16.17.1+
  - odkaz na stiahnutie: <https://nodejs.org/en/download>
- nainštalovaný Camunda modeler
  - odkaz na stiahnutie: <https://camunda.com/download/modeler/>

## Postup spustenia Self-managed verzie Camundy

1. stiahneme repozitár (<https://github.com/camunda/camunda-platform/tree/8.2.4>)
2. v príkazovej riadke (PowerShell, Terminal) sa presunieme do umiestnenie projektu
3. spustíme príkaz `docker compose -f docker-compose-core.yaml up`, ktorým sa spustí stahovanie docker kontajnerov
4. po skončení stahovania skontroluje v rozhraní Docker Desktop, či sa vytvorili a bežia kontajnere `zeebe`, `operate`, `tasklist` a `elasticsearch`
5. prekontrolujeme adresy rozhraní Tasklist a Operate (<http://localhost:8081/> a <http://localhost:8081/>)

## Postup nasadenia diagramov

1. stiahneme repozitár (<https://github.com/marcel-zec/masters-thesis>)
2. v Camunda Modeler otvoríme všetky procesy z repozitára na adrese `camundaProject/bpmn` a DMN diagrami na adrese `camundaProject/dmn`
3. nasadíme diagrami pomocou tlačidla nasadenia

## Postup spustenia serveru

1. v stiahnutom repozitáry (<https://github.com/marcel-zec/masters-thesis>)
2. v príkazovej riadke (PowerShell, Terminal) sa presunieme do priečinku `server`
3. spustíme príkaz `npm install`, ktorý nainštaluje potrebné závislosti projektu
4. po nainštalovaní spustíme príkaz `node index.js`, ktorý spustí server
  - v termináli sa zobrazí správa `Application started and listening on port 3000`
5. na adrese <http://localhost:3000/> otvoríme webové rozhranie