TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Řízení dopravníkových systémů pomocí rozhodovacích tabulek

Diplomová práce

Studijní program:	N2612 Elektrotechnika a informatika
Studijní obor:	Mechatronika
Autor práce:	**Bc. Jakub Sokola**
Vedoucí práce:	Ing. Tomáš Martinec, Ph.D. Ústav mechatroniky a technické informatiky

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání diplomové práce

Řízení dopravníkových systémů pomocí rozhodovacích tabulek

Jméno a příjmení: **Bc. Jakub Sokola**
Osobní číslo: M20000187
Studijní program: N2612 Elektrotechnika a informatika
Studijní obor: Mechatronika
Zadávající katedra: Ústav mechatroniky a technické informatiky
Akademický rok: **2021/2022**

Zásady pro vypracování:

1. Seznamte se se simulačním SW Tecnomatix Plant Simulation od firmy Siemens.
2. Proveďte rešerši existujících konvencí pro zápis algoritmu pomocí rozhodovacích tabulek, na jejímž základě navrhněte a popište vlastní způsob definování jednoduchých řídících algoritmů formou rozhodovacích tabulek, který by dokázal přehledněji strukturovat řídicí algoritmy dopravníkové technologie v simulačních modelech.
3. Vytvořte parametrizační prvek v prostředí SW Plant Simulation, který bude schopen řídit simulační model pomocí algoritmů zapsaných formou rozhodovací tabulky z bodu 2.
4. Aplikujte parametrizační prvek z bodu 3 na konkrétní simulační úlohu (dopravníkový uzel) a popište výhody a nevýhody řízení modelu pomocí rozhodovacích tabulek.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	40–50 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] TECNOMATIX PLANT SIMULATION [online]. [cit. 2021-03-06]. Dostupné z: https://www.axiomtech.cz/25357-texnomatix-plant-simulation

[2] VOLF, Luděk, Libor BERÁNEK a Petr MIKEŠ. Počítačová simulace ve strojírenské výrobě [online]. [cit. 2021-03-06]. ISBN 978-80-7043-934-0. Dostupné z: https://dspace5.zcu.cz/bitstream/11025/16403/1/Volf.pdf

[3] Plant Simulation: interní materiály oddělení Servisu plánování. Škoda Auto a. s.

[4] ŠTOČEK, Jiří. Optimalizace materiálového toku ve vybraném průmyslovém závodě. Brno, 2004. Disertační práce. Vysoké učení technické v Brně.

[5] SIEMENS: Plant Simulation & Throughput Optimization [online]. [cit. 2021-03-06]. Dostupné z: https://www.plm.automation.siemens.com/global/en/products/manufacturing-planning/plant-simulation-throughput-optimization.html

Vedoucí práce:	Ing. Tomáš Martinec, Ph.D. Ústav mechatroniky a technické informatiky
Datum zadání práce:	12. října 2021
Předpokládaný termín odevzdání:	16. května 2022

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Josef Černohorský, Ph.D.
vedoucí ústavu

V Liberci dne 12. října 2021

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

1. května 2022 Bc. Jakub Sokola

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Tomáši Martincovi, Ph.D., který po celou dobu mého vysokoškolského studia vedl moje akademické práce. Vždy mi dával cenné rady a podněty k daným tématům.

Dále chci poděkovat kolegům z oddělení simulačních studií ve společnosti Škoda Auto a.s. za konstruktivní rady a návrhy při tvorbě této diplomové práce, vytvoření přátelského pracovního prostředí a vstřícnost v pracovních i osobních záležitostech.

V neposlední řadě bych rád poděkoval své rodině za neustálou podporu, motivaci, trpělivost i rozptýlení při studiu.

Anotace

Práce je zaměřena na tvorbu řídicích algoritmů pro simulační modely v softwaru Plant Simulation. Čtenáře seznamuje s dopravníkovou technikou, terminologií počítačové simulace, základními prvky softwaru a jejich řízením. Následně je navržen nový způsob vytváření programových kódů pro řízení pomocí rozhodovací tabulky. V praktické části je popsána její struktura, deklarace a implementace jakožto parametrizačního a řídicího prvku do projektů v prostředí Plant Simulation, a to na konkrétním příkladu. V závěru práce jsou shrnuty přínosy řízení pomocí vytvořeného prvku oproti dosud využívanému řízení standardním programováním.

Klíčová slova

rozhodovací tabulka, algoritmus, řídicí logika, počítačová simulace

Annotation

The work is focused on the creation of control algorithms for computer simulation models in Plant Simulation software. Reader is acquainted with conveyor technology, computer simulation terminology, basic software objects and their control. Subsequently a new way of defining control codes proposed through the decision table. In practical part is describtion of its structure, declaration and implementation as parameterization and control object into projects in the Plant Simulation environment on concrete example. In conclusion are summarized the benefits of control by the created parameterization element compared to the previously used program code control.

Key Words

decision table, algorithm, control logic, computer simulation

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

DP – diplomová práce

SW – software

VW – koncern Volkswagen

PS – Plant Simulation

VDA – (Verband der Automobilindustrie) – sdružení automobilového průmyslu

MU – (Moveable Unit) - pohybující se objekt, výrobek

ET – (Entscheidungstabelle) – rozhodovací tabulka

Úvod

V průmyslových závodech se při plánování výrobních systémů stává běžnou praxí sdílení projektových úkolů v týmu. Stejně tak tomu je v oblasti simulačních studií, které slouží jako podpůrné rozhodovací nástroje pro plánování nových, či optimalizaci stávajících výrobních oblastí. Zvyšující se komplexita projektů často vyžaduje spolupráci více řešitelů na jediném simulačním modelu. Aby byla zaručena shoda objektů popisujících zkoumanou oblast, jsou pro simulační studie zavedeny různé standardy. Příkladem je zavedení rozsáhlého standardu VDA, který sjednocuje objekty simulačního modelu napříč automobilovými koncerny. Problém však stále přetrvává v možnostech vytváření algoritmů (metod), které odrážejí logiku chování řešené oblasti.

Z důvodu chybějícího obecného standardu si každý řešitel volí vlastní přístup k tvorbě programových algoritmů. Dílčí kroky přístupu mohou být odlišné a sdílení řídicích algoritmů se stává obtížným. Minimálně v rámci pracovního týmu by měl být definován standard, který by řešitelé dodržovali.

Cílem diplomové práce je navrhnout nástroj pro standardizovanou tvorbu řídicích algoritmů, a to formou rozhodovací tabulky. Návrh by měl být dostatečně obecný, aby bylo možné nástroj aplikovat pro simulační studie různých oblastí. Nástroj bude vytvořen ve formě parametrizačního prvku v prostředí Plant Simulation, ke kterému budou mít řešitelé v týmu přístup. Využití rozhodovací tabulky pro definování řídicích algoritmů řešených oblastí by umožňovalo efektivněji tvořit simulační modely, eliminovat chyby vzniklé v logice řízení již při vzniku modelu a díky zpětné analýze programového algoritmu efektivně dekomponovat logiku řízení.

Rizikem při definování logiky řízení simulačního modelu pomocí parametrizačního prvku by mohlo být snížení výpočetního výkonu, který se přímo odráží v době trvání simulačního běhu. Doba trvání simulačního experimentu je klíčová, neboť simulační čas je násoben počtem prováděných experimentů (v praxi se může jednat i o stovky experimentů s jedním simulačním modelem).

Teoretická část

1. Základní terminologie

Systém je soubor prvků se vzájemnými vazbami, který jako celek má určitý vztah ke svému okolí. Každý systém lze charakterizovat dvěma základními vlastnostmi: [1]

- Chováním – je závislost, která za působení určitých okolních podnětů určuje vztah mezi vstupy a výstupy systému
- Strukturou – určuje uspořádání mezi prvky daného systému a chování těchto prvků

V rámci této diplomové práce bude pod pojmem systém uvažováno seskupení prvků přepravní technologie pro přepravu a manipulaci s materiálem.

Model je zjednodušená náhrada reálného systému, která je závislá na fázi projektu, dostupnosti vstupních dat, domluvy se zadavateli studie, atd. Zahrnuje především informace, které mají vliv na řešenou oblast, a slouží hlavně ke zkoumání chování reálného systémů. Modely mohou být fyzické nebo matematické, statické nebo dynamické, deterministické nebo stochastické a diskrétní nebo spojité. [2]

- Fyzický – zjednodušené makety systému
- Matematické – systém je reprezentován rovnicemi, které popisují jeho chování
- Statické – popisují konkrétní události systému v danou chvíli
- Dynamické – popisují chování systému v daném časovém období
- Deterministické modely – výstup systému je ovlivněn pouze jeho vstupy
- Stochastické modely – výstup systému je ovlivňován vstupy a náhodnými veličinami
- **Diskrétní modely** – vstupy a výstupy se mění skokově
- Spojité modely – vstupy a výstupy se mění kontinuálně s časem

Tato práce se zaměřuje výhradně na diskrétní modely.

Materiálový tok je pohyb materiálu ve výrobním procesu nebo v oběhu, prováděný pomocí manipulačních, přepravních a pomocných prostředků cílevědomě tak, aby materiál byl k dispozici na daném místě a v potřebném množství, nepoškozený v požadovaném okamžiku, a to s předem určenou spolehlivostí. [5]

Prvky materiálového toku lze rozdělit do dvou základních skupin dle jejich využití v rámci materiálového toku: [3]

- Pasivní prvky – objekty, které jsou přepravovány v rámci zkoumané oblasti. Může se jednat o krabice, palety, díly, obaly, odpad, atd.
- Aktivní prvky – realizují tok pasivních prvků. Uskutečňují logické (netechnologické) operace s pasivními prvky, jako např.: balení, nakládka, přeprava, uskladnění, atd.

Optimalizace materiálového toku se snaží eliminovat náklady spojené s nevhodným rozmístěním výrobních objektů, skladovacích ploch, uspořádáním pracovních míst, nevhodnou volbou navržení manipulační či přepravní technologie, atd. Správné navržení materiálového toku by mělo zahrnovat: [3]

- Jednotný směr toku bez křížení a zpětného pohybu materiálu
- Eliminaci zbytečné manipulace s materiálem
- Nejkratší možnou transportní vzdálenost
- Uspořádání výrobních prostředků pro možnou flexibilitu v rámci výrobních požadavků
- Zajištění rytmičnosti, nepřetržitosti a plynulosti materiálu
- Zvýšení mechanizace při manipulaci s materiálem pro zajištění růstu práce
- Vytvoření vhodných pracovních a bezpečných podmínek při manipulaci s materiálem

Simulace je imitace zkoumaného procesu nebo systému i s jeho dynamickými procesy. Umožňuje pozorovat chování zkoumané oblasti za účelem vyvození závěrů týkajících se jejího chování. [2]

Experiment je postup, při kterém se zkoumají určité jevy při různých podmínkách. Slouží pro získání nových poznatků. [4] Pro simulační modely slouží ke vzájemnému porovnání výstupů různých variant při totožné změně vstupů.

Optimalizací systému je myšleno nalezení uspokojujícího řešení vzhledem k existujícím omezení. Často je v rámci optimalizace zkoumáno několik variant řešení, na které může mít vliv i finanční nákladnost realizace. Vybraná varianta nemusí být optimální, ale v rámci definovaných omezení se k optimální variantě přibližuje. [3]

Informační tok je pohyb informací. Pohybem materiálu jsou vytvářeny informace, které mohou ovlivňovat další pohyb, vytvoření či usměrnění materiálu. Zpracování informací má vést k organizovanému, plynulému a cílevědomému řízení materiálového toku a jeho zpětné kontrole. Jeden bez druhého nemůžou fungovat. [4]

2. Přepravní technologie

Následující kapitola popisuje základní druhy přepravní technologie, která ovlivňuje materiálový a informační tok. Je hojně používaná v různých oblastech. Operace přepravní technologie lze rozdělit do několika základních skupin [3]:

- Mezioperační přeprava – přemístění mezi technologickými či kompletačními pracovišti, místy kontroly nebo skladováním
- Skladové operace – uskladnění či vyskladnění
- Meziobjektová přeprava – např. přeprava mezi lakovnou a montáží

Přepravní technologie je v automobilových průmyslech používána k přesunu karoserií. Lze ji členit do několika skupin podle způsobu přepravy karoserie. V automobilových závodech jsou nejvíce používané následující dvě [3]:

- Skidová technika – přepravovaná karoserie je umístěna na speciálním podstavci, tzv. skidu. Převážně se jedná o pozemní přepravu karoserií.
- Závěsová technika – karoserie je uchycena závěsem, který má vlastní pohonnou jednotku a pohybuje se po kolejnici.

Prvky přepravní technologie mohou mít na své konstrukci připevněny senzory pro identifikaci pozic karoserií, kontroly svárů, dojezdy, atd. Senzory vysílají signály nadřazenému systému, který řídí pohonné jednotky přepravní technologie vhodným způsobem, tak, aby nedošlo ke kolizi materiálu a zároveň byl materiál v požadovaném množství včas na určeném místě.

Následně bude popsána struktura a princip činnosti pouze skidové přepravní technologie, která je dále zmiňována v této DP.

2.1 Dopravníky

Přepravují materiál po definované dráze, která je dána uspořádáním dopravníkové trati. Fyzická realizace dopravníku může být různorodá. V automobilovém průmyslu je používána válečková struktura dopravníku, která se skládá z válečkových drah. Válečková dráha je segment, který má vlastní pohonnou jednotku, případně řízenou frekvenčním měničem, pro řízení rychlosti přepravy karoserie. Pokud délka válečkové dráhy neodpovídá délce skidu, tak může být nastavena podpěrnou kladkou (Obr. 1) [3]

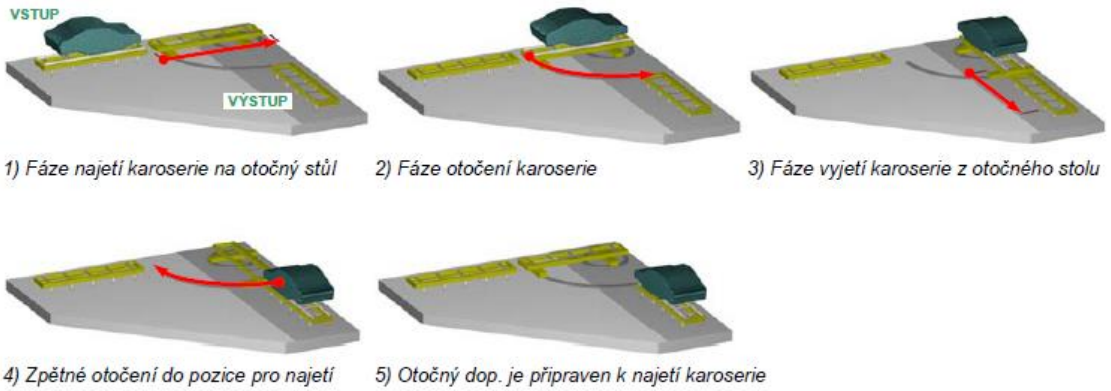
Obr. 1 – Válečkový dopravník [3]

2.2 Kyvné a otočné stoly

Slouží ke změně směru materiálového toku, který je měněn mezi předcházejícími a navazujícími dopravníky. Karoserie přepravená na stůl je natočena o definovaný úhel ve směru, kterým má být přepravována dále.

2.2.1 Kyvné stoly

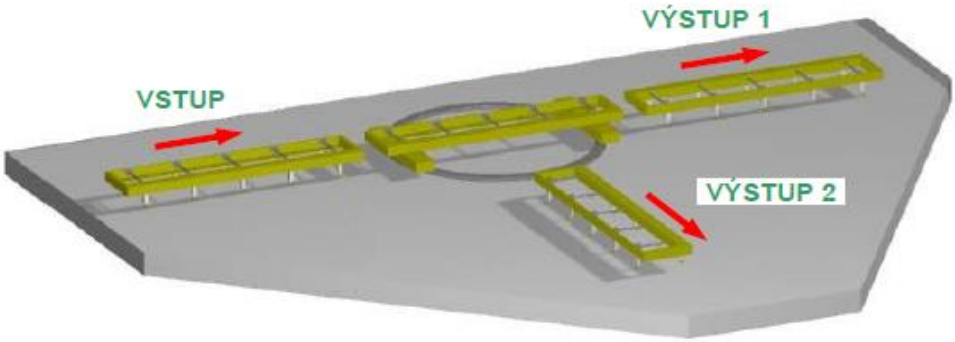
Kyvné stoly mají excentrický střed otáčení a zajišťují natočení karoserie o 10 až 90°. [3] Na následujícím Obr. 2 je znázorněno natočení kyvného stolu o 90° v jednotlivých fázích otočení.



Obr. 2 – Kyvný stůl [3]

2.2.2 Otočné stoly

Otočné stoly mají osu rotace ve svém středu a mohou se otáčet o 0 až 360° v obou směrech. [6] Následující obrázek (Obr. 3) zobrazuje otočný stůl s jedním vstupním dopravníkem a dvěma výstupními.

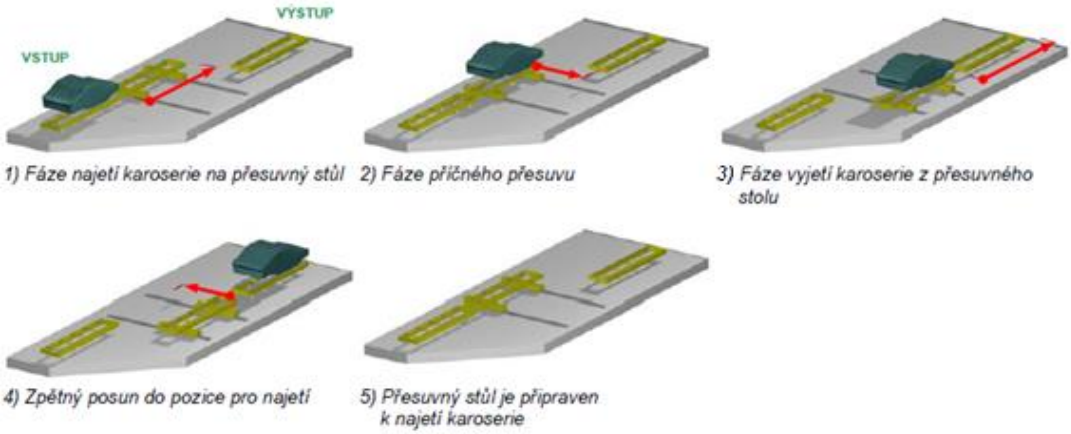


Obr. 3 – Otočný stůl [3]

2.3 Příčný přesuvný dopravník

Přesouvá karoserie mezi příčně uloženými dopravníky. Je možné propojit dopravníky s více vstupy a výstupy.

Na obrázku (Obr. 4) je znázorněn přesuv karoserie v jednotlivých fázích.



Obr. 4 – Příčně přesuvný dopravník [3]

2.4 Příčný pásový dopravník

Konstrukce příčně pásového dopravníku obsahuje zvedací stoly a dva pásové dopravníky. Přesuv karoserií mezi zvedacími stoly je realizován pásovým dopravníkem. Zvedací stoly slouží pro najetí či vyjetí karoserie na příčně pásový dopravník nebo spouštění či zdvihání karoserií z pásového dopravníku. V principu příčný přesuvný dopravník a příčný pásový dopravník (se dvěma zvedacími stoly) plní stejnou funkci. Jedním z rozdílů je, že v případě zablokování výstupu může být příčný pásový dopravník obsazen několika karoseriemi, jejichž počet odpovídá počtu zvedacích stolů. Oproti příčně přesuvnému dopravníku tím zvyšuje možné zaskladnění karoserií na malém prostoru. [3] [6]

Na Obr. 5 je zobrazen příčně pásového dopravníku se dvěma zvedacími stoly v jednotlivých fázích přesuvu karoserie.

Obr. 5 – Příčně přesuvný stůl [3]

3. Počítačová simulace

Na základě definice simulace v kapitole 1 je chápána počítačová simulace jako výzkumná technika, jejímž cílem je zobrazit reálný systém v počítačovém virtuálním prostředí, kde lze provádět simulační experimenty pro získání informací o přibližném chování skutečného systému, bez ovlivnění či existence reálného systému. [1]

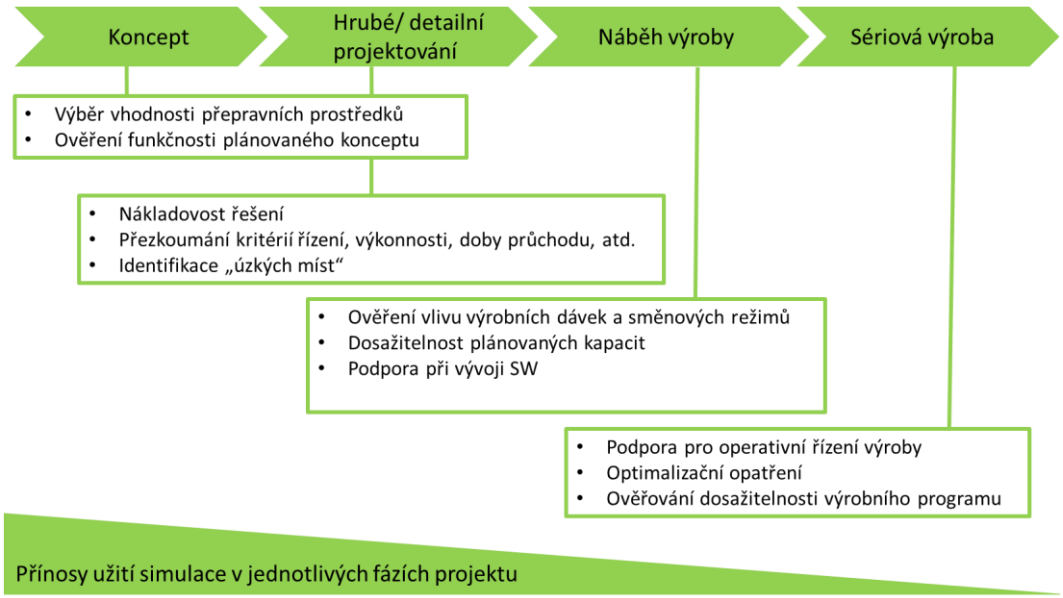
Počítačová simulace je vhodná, někdy i jediná, možnost pro analýzu chování zkoumaných systémů. Umožňuje značnou flexibilitu v testování systémů pro různé scénáře, jejichž podoba se může, často i dynamicky, měnit dle požadavků firmy. Relativně rychle lze chování zkoumaných systémů odhadnout na základě chování simulačního modelu, aniž by zkoumaný systém byl fyzicky k dispozici. [1] [5] Simulační nástroje jsou využívány v různých oblastech pomocí simulačních softwarů, jako např. Matlab, Ansys, Simpro, Plant Simulation, atd.

V automobilových závodech se stal hlavním předmětem zájmu materiálový tok, jehož optimalizace se v této oblasti stává běžnou úlohou, ale bez využití simulačních nástrojů je téměř nemožné dosáhnout optimálního materiálového toku. Postupným využíváním simulačních studií v automobilovém průmyslu vznikla potřebná standardizace tvorby a správy simulačních modelů a sjednocení pravidel provádění experimentů. V rámci koncernu VW je směrodatná pracovní skupina VDA, která standardizuje rozhraní a SW nástroje pro simulace. [7]

Simulaci je vhodné použít, pokud nelze pro popis zkoumaného systému použít exaktního analytického matematického modelu, např. vzhledem ke stupni složitosti výrobního procesu, šířce výrobního programu a vysokému stupni proměnných. [8]

3.1 Využití simulace

Využití simulačních nástrojů je závislé na fázi projektu či řízení výroby. Slouží k ověření funkceschopnosti konceptu před samotným náběhem výroby, ověření kvality řízení, výkonnosti systému, stavu zásob, atd. [7] Využití simulace v jednotlivých fázích projektu je znázorněno na následujícím Obr. 6.



Obr. 6 – Využití simulace v jednotlivých fázích projektu [7]

Největší přínos má využití simulace ve fázi konceptu, kde doposud vynaložené prostředky na realizaci projektu jsou minimální. Návrhem ideálního stavu systému vzhledem k finančním, prostorovým, technickým a dalším omezením lze značně snížit nákladnost realizace. Ve fázi hrubého projektování je koncepční model zpřesňován a ve fázi detailního projektování může být testována logika řízení, výkonnost, doba průchodu, atd. V sériové výrobě jsou simulační nástroje vhodné při analýze úzkých míst, zlepšování stávajících procesů, posuzování změn technologie nebo výrobního programu, atd. [4][5]

Při uvážení rychlosti, nákladů, zaškolení a možnosti provádění experimentů bez zásahu do reálného systému je simulace silným nástrojem pro zkoumání dané oblasti. Lze testovat různé varianty systému v bezrizikovém prostředí s velkou mírou vizualizace důsledků. Nejčastěji řešené problémy jsou: [3]

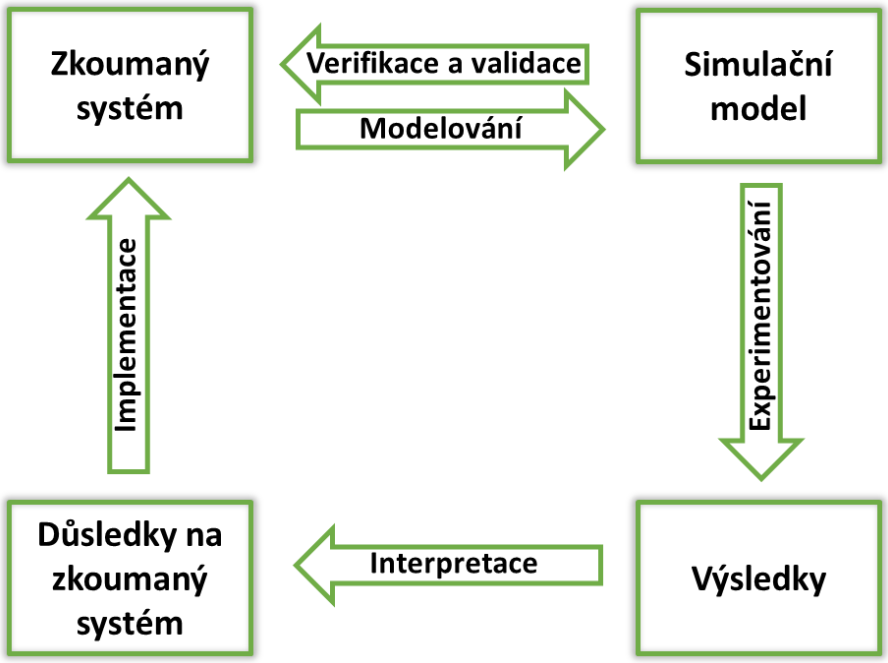
- Identifikace úzkých míst ve výrobě
- Zkracování průběžné doby výroby
- Optimalizace výrobních dávek
- Plánování kapacit
- Projektování výrobních celků
- Optimalizace požadavků na pracovní síly
- Podpora při vývoji a testování řídicího softwaru
- Školení personálu

Ačkoli je simulace vhodná pro řešení různých typů problému, je nutné zvážit aspekty, které se s využitím simulace pojí. Ty se mohou týkat finanční náročnosti nebo vhodnosti jejího použití. [2][4]

- Nutnost vynaložení finančních prostředků na simulační SW
- Simulace může být nákladnější než reálný experiment
- Nákladnost simulace nemusí překročit ušetřené náklady
- Může vyžadovat vytvoření většího pracovního týmu

3.2 Simulační model

Simulační model je systém, který může nabývat několika podob v závislosti na simulované oblasti, požadovaném cíli simulace a systémových hranicích. Systémové hranice určují oblast zájmu zkoumané oblasti dle požadovaného cíle simulační studie a vymezují vstupy a výstupy reálného systému. [4] [1] Proces, kterým z reálného systému získáme simulační model, se nazývá modelování a cyklus simulační studie je naznačena na Obr. 7. [8]



Obr. 7 – Cyklus simulační studie [3]

Získaný simulační model je verifikován a validován. Tyto dva úkony slouží k ověření správné aproximace reálného systému v předem definované míře detailnosti. Pokud se

simulační model a reálný systém v definované míře neshodují, je simulační model zpřesňován až k požadované míře shody. [1][2]

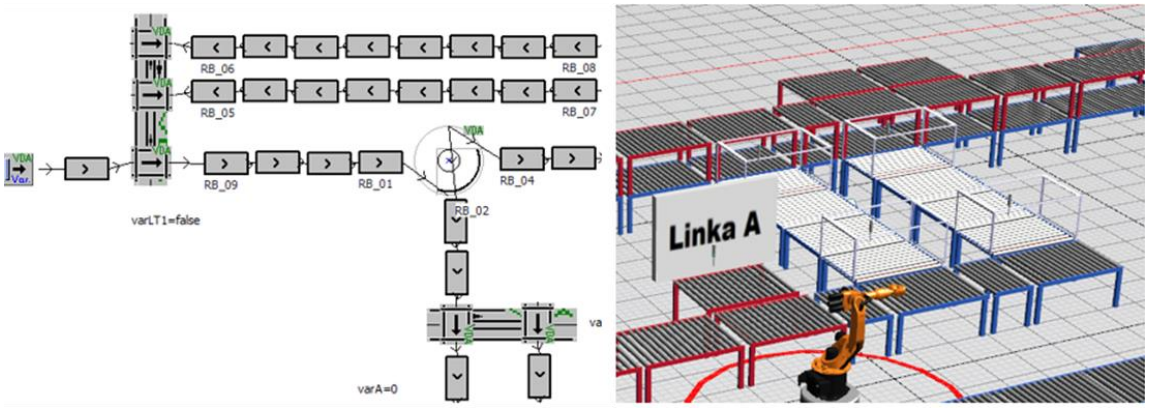
- Verifikace se zabývá tím, zda simulační model správně reflektuje reálný systém. Porovnává především strukturu, komponenty, parametry, atd. simulačního modelu a zkoumaného systému, které se musí v definované míře shodovat. [2]
- Validace se zabývá tím, zda je reálný systém správně reflektován simulačním modelem. Postupným zpřesňováním se simulační model stále více přibližuje reálnému systému. [2]

Pokud chování systémů bude v definované míře shodné, tak lze simulační model prohlásit za verifikovaný a validovaný.

3.3 Plant Simulation

Jedná se o simulační SW vyvinutý společností Siemens PLM Software, který slouží pro modelování, simulaci a analýzu výrobních, dopravních, manipulačních a dalších systémů. Zejména se používá pro zobrazení výrobních procesů v počítačovém virtuálním prostředí a jejich následnou optimalizaci. Pomocí počítačového modelu lze provádět experimenty například změnou technologie, logiky řízení, výrobní dávky a dalších bez dopadu na výrobu. K vyhodnocení výsledků simulačního modelu s různými scénáři se používají statistické a grafické prvky, které porovnávají průchodnost, obsazenost, poruchy a další veličiny popisující chování systému. [9]

Plant Simulation (dále jen PS) umožňuje tvořit simulační modely ve 2D nebo 3D (viz Obr. 8).



Obr. 8 – Příklad 2D a 3D modelu

3.3.1 Prvky Plant Simulation

Při vytváření modelu je k dispozici knihovna tříd, která obsahuje prvky, jejich grafiku a parametry. Pro vytváření modelů nabízí PS velké množství prvků, které lze podle funkce zařadit do skupin materiálového toku, informačního toku, zdrojů, uživatelského rozhraní, pohyblivých objektů, atd. Následně budou popsány skupiny, které se nejčastěji používají pro simulační modely automobilových závodů. U každé skupiny je uvedeno několik prvků s jejich 2D ikonami a krátkým popisem. [5][10]

Základní prvky materiálového toku:

- Ovladač událostí (Event controller) – ovládá délku, rychlost a konec simulačního běhu, atd.
- Stanice (Station) – představuje pracoviště (např. robotickou buňku, soustruh, manipulátor), u kterých lze definovat čas operace (takt), poruchovost, atd.
- Zdroj (Source) – zdroj materiálu vstupujícího do sledované oblasti
- Spojka (Connector) – propojuje jednotlivé prvky simulačního modelu
- Rozdělovač (Flow Control) – řídí směr materiálového toku
- Dopravník (Conveyor) – přepravuje materiál
- Výstup (Drain) – zánik materiálu na konci sledované oblasti, což může obsahovat navazující procesy

Základní prvky informačního toku:

- Tabulka (Table) – zápis informací v tabulkové formě libovolného datového typu (string, object, integer, atd.)
- Proměnná (Variable) – zápis informace libovolného datového typu
- Metoda (Method) – psaní logiky řízení pomocí programového kódu

Základní prvky uživatelského rozhraní:

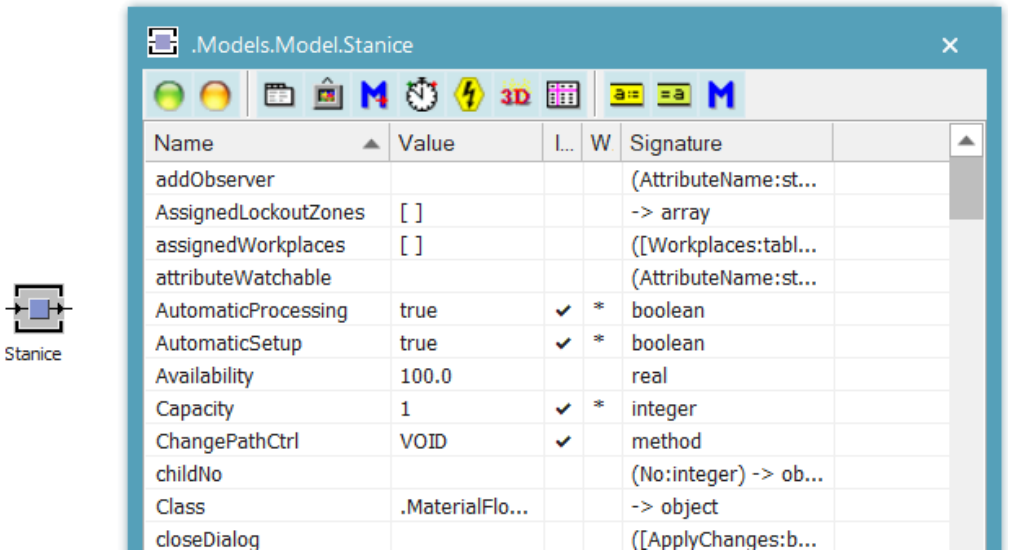
- Komentář (Comment) – uživatelské komentáře
- Graf (Chart) – zobrazuje data
- Tlačítko (Button) – ovládací prvek

Základní prvky pohyblivých objektů (movable unit – MU):

- Díl (Part) – pasivní materiál
- Paleta (Container) – pasivní materiál, který může obsahovat díly

Jednotlivé prvky jsou specifikovány svými atributy konkrétních datových typů (integer, real, boolean, atd.). Atributy jsou vedené v přehledné tabulce a v rámci řízení simulačních modelů je možné se na ně odkazovat nebo měnit jejich hodnotu, pokud však nejsou určené pouze pro čtení. Vedle pevně definovaných atributů lze přidávat vlastní atributy, které je nutno definovat názvem a datovým typem. [6]

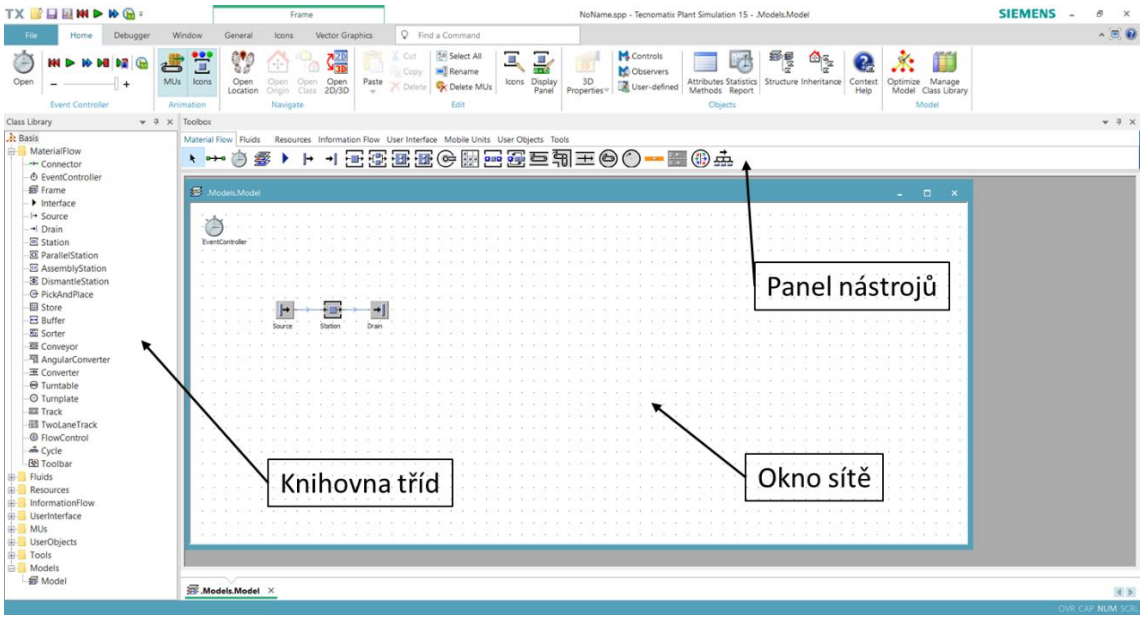
Na následujícím Obr. 9 je zobrazena tabulka atributů prvku *Stanice*.



Obr. 9 – Atributy prvku Stanice

3.3.2 Pracovní prostředí Plant Simulation

Základní zobrazení PS obsahuje pracovní okno pro tvorbu simulačních modelů, které bude vypadat obdobně jako na Obr. 10.



Obr. 10 – Pracovní okno Plant Simulation

Knihovna tříd obsahuje veškeré prvky pro tvorbu modelu. Prvky jsou uspořádány do složek dle svých funkcí. Lze vytvářet vlastní složky i prvky dle potřeby uživatele. [10]

Pomocí panelu nástrojů lze rychle přistoupit k často používaným prvkům, které jsou dostupné i v knihovně tříd. Panel nástrojů obsahuje záložky s příslušnými prvky dle jejich funkcí a lze také přidávat vlastní záložky s vlastními prvky. [10]

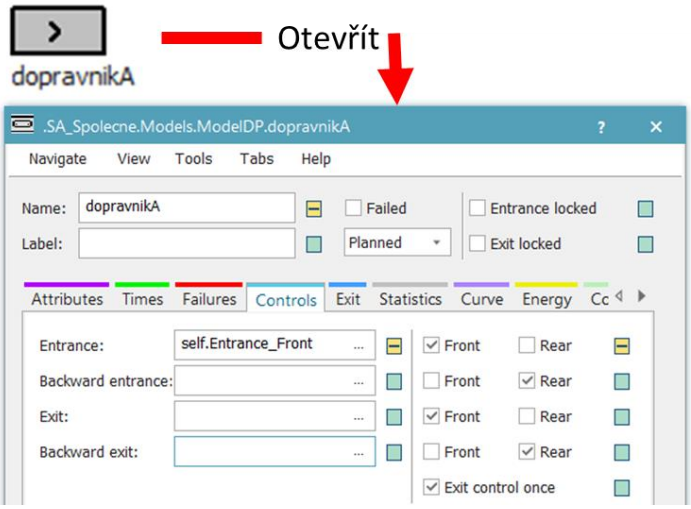
Okno sítě je hlavním pracovním oknem a slouží pro umísťování prvků při tvorbě modelu. Uživatel může měnit rastr okna, nahrát pozadí (např. obrázek v podporovaném formátu), vytvářet vlastní grafiku, atd. (např. popisky na pozadí). [10]

3.3.3 Logika řízení simulačních modelů v PS

Logika řízení je nejčastěji definována programovým kódem, k tomu PS využívá vlastní programovací jazyk SimTalk. Při tvorbě kódu je využíváno podmínek stylu *if…else…end, if…elseif…else…end, for…loop, and, or, end*, atd. V rámci programu lze pracovat s atributy prvků simulačního modelu. Nápověda SimTalk kompilátoru pomáhá definovat správnou adresu těchto atributů, kontroluje jejich existenci, správnost datových

typů, atd. [10] Samotný programovací jazyk je objektově orientovaný a syntaxí podobný jazykům, které jsou založeny na bázi jazyka C (tzn. C++, C#, Java, atd.).

Prvky simulačního modelu mají prostor pro definování metod, které můžou být volány při různých událostech (např.: vstupu, výstupu, přejezdu MU, najetí na senzor, atd.). Tyto události (impulsy) jsou obrazem impulsů ze senzorů přepravní technologie. Pro aktivaci senzoru v simulačním modelu je nutno vytvořit metodu a definovat ji požadovanému senzoru v záložce *Controls* daného prvku viz Obr. 11, kde senzoru *Entrance* byla přiřazena metoda *Entrance_Front*. U jednotlivých senzorů lze pomocí zaškrtávacích polí volit, zda bude metoda senzoru volána najetím (*Front*) nebo opuštěním (*Rear*) MU.



Obr. 11 – Definice senzoru

4. Algoritmy

Algoritmus je jednoznačný srozumitelný postup definovaný v konečném počtu příkazů, jak na základě vstupních údajů vyřešit danou úlohu a získat údaje výstupní. [12]

Vývoj algoritmu probíhá v několika etapách: [12]

- Rozbor problému – úplné a jednoznačné stanovení řešeného problému
- Vytvoření algoritmu – návrh řešení daného problému, který by měl být:
    - Jednoznačný – vždy jednoznačně určovat následující činnost
    - Konečný – postup skončí v konečném čase
    - Všeobecný – použitelný pro celou množinu vstupních údajů
    - Opakovatelný – při stejném vstupu musí být i stejný výstup
    - Srozumitelný – přehledný
- Implementace – realizace navrženého algoritmu
- Testování a ladění – prověření funkčnosti a odstranění chyb

Algoritmy lze vytvořit v několika podobách: [12]

- Graficky – vývojové diagramy obsahující rozhodovací podmínky a akce
- Slovně – popis udávající pořadí určitých akcí, které vedou k tíženému cíli
- Matematicky – řešení soustav rovnic, determinanty – postupy výpočtu
- Programovým kódem – v technické praxi je nejrozšířenější forma zápisu algoritmu. Je tvořen rozhodovacími podmínkami a akcemi zapsanými dle syntaxe příslušného programovacího jazyka

Všechny tyto druhy zápisu algoritmu lze použít pro definování řešení jakéhokoli problému, omezující je implementace některých zápisů přímo do požadovaného pracovního prostředí. PS umožňuje psát řídicí algoritmy pouze ve formě programového kódu, čímž lze definovat jakoukoli úlohu, ale postrádá přehlednost jiných forem zápisu algoritmu.

4.1 Rozhodovací tabulky

Jsou přehledným nástrojem pro vyhodnocení určitého problému. Využívají se v širokém spektru oblastí od účetnictví, personalistiky, managementu, atd. V technické praxi však stále převládá vyhodnocování problémů pomocí programových kódů.

Rozhodovací tabulka má několik základních oblastí: [13]

- Podmínky – jsou definované v řádcích či sloupcích, záleží na typu dané tabulky. Pro zápis jsou používány logické operátory <, >, =, atd.
- Pravidla – slouží pro vyhodnocení podmínek. Používají dvoustavové logické operátory 0, 1, Y, N, atd.
- Akce – opatření při splnění daných podmínek

Nástroj rozhodovacích tabulek je využíván především v oblasti managementu a byznys plánování. Software *Visual Paradigm*, který se používá právě v těchto oblastech, má implementované rozhodovací tabulky i s uživatelským rozhraním s funkcemi: přidání, odebrání, reorganizace nebo mazání podmínky či akcí. Tabulka v tomto SW může mít následující podobu (Tab. 1). [14]

Tab. 1 – Příklad rozhodovací tabulky (Visual Paradigm) [14]

	Pravidla					
Podmínky	**1**	**2**	**3**	**4**	**5**	**6**
P1. **kojenci (<2)**	Y	Y				
P2. **mládež (>2 a <16)**			Y			
P3. **domácí lety**	Y				Y	
P4. **mezinárodní lety**		Y				Y
P5. **brzká rezervace**				Y	Y	
P6. **mimosezónní lety**						Y
Akce	**1**	**2**	**3**	**4**	**5**	**6**
A1. **sleva 10 %**			Y	Y		
A2. **sleva 15 %**						Y
A3. **sleva 20 %**					Y	
A4. **sleva 70 %**		Y				
A5. **sleva 80 %**	Y					

Tento příklad použití rozhodovací tabulky udává procentuální slevu ceny letenek v závislosti na destinaci, věku cestujícího a období. Využívá tři sekce: podmínky, akce a

pravidla. Pomocí značky *Y* (*YES*) nebo *N* (*NO*) u jednotlivých podmínek je určena akce, která se má vykonat. [14]

Pro lepší pochopení orientace v rozhodovací tabulce je uveden následující příklad. Dospělý muž letí s kojencem mimo sezónu do jiného státu. Dle uvedené tabulky budou slevy z ceny letenek následující:

- Dospělý muž – splněný podmínkový sloupec 6 (P4, P6) – sleva 15 % (A2)
- Kojenec – splněný podmínkový sloupec 2 (P1, P4) – sleva 70 % (A4)
  – splněný podmínkový sloupec 6 (P4, P6) – sleva 15 % (A2)

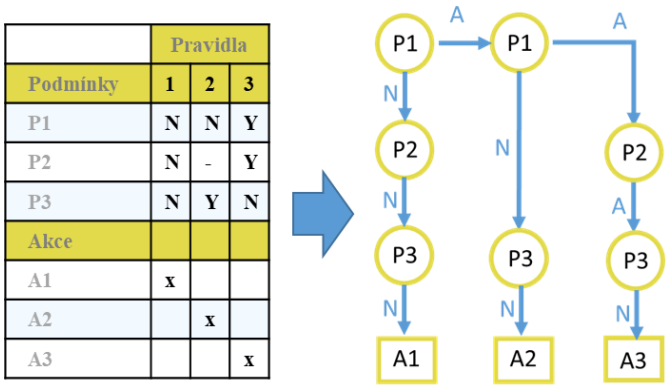
Jedná se o velmi intuitivní rozhodovací algoritmus uplatňování slev na letenky. Snahou této diplomové práce je rozvinout tuto myšlenku jednoduchého zápisu algoritmu na úroveň definování řídicích logik pro simulační modely v prostředí PS. Logiky řízení by mohly být zapisovány pomocí rozhodovací tabulky, která bude zohledňovat potřeby simulačních modelů a kompatibilitu s prostředím PS.

4.1.1 Zpracování rozhodovací tabulky

Rozhodovací tabulky jsou snadným efektivním a rychlým nástrojem pro orientaci daného problému, ale počítačový program nepracuje přímo s nadefinovanou tabulkou. Počítač zpracovává instrukce (programový kód), který je nutné z rozhodovací tabulky získat. Převod z rozhodovací tabulky na programový kód je v této práci označován jako transformace a je realizován transformačním algoritmem, který je hodnocen z několika hledisek: [15]

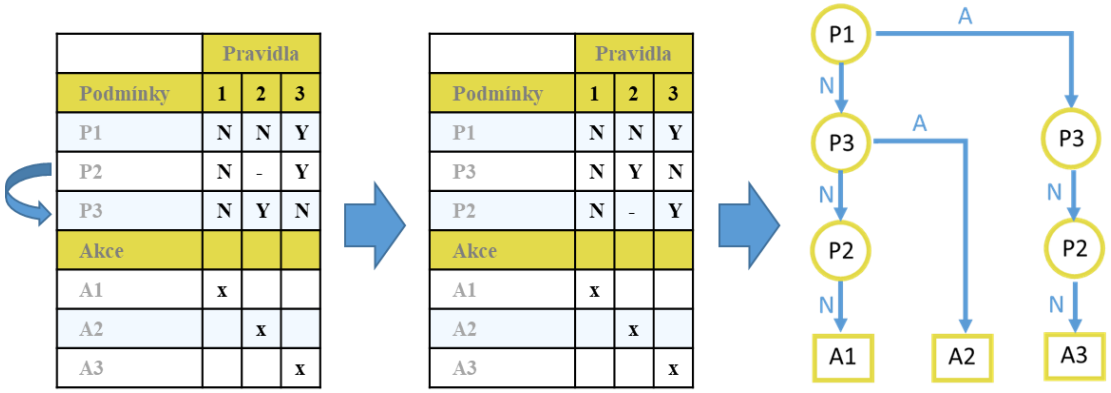
- **Čas transformace** – je ovlivněn velikostí tabulky a efektivitou transformačního algoritmu. Je nutno jej zohledňovat, pokud by ovlivňoval řešení úlohy. V případě simulačních experimentů je transformace prováděna před samotným simulačním během, tudíž je časově zatěžovat nebude.
- **Efektivita programového kódu** – z hlediska výpočetní náročnosti simulačních experimentů je efektivita zpracování programového kódu klíčová. Zpracování závisí na počtu rozhodovacích podmínek, které se v kódu vyskytují. Transformace, která je provedena bez předzpracování (srovnání) rozhodovací tabulky, vede k většímu počtu rozhodovacích podmínek, tedy k neefektivnímu programovému kódu, [15][16] viz Obr. 12, kde je naznačena transformace

rozhodovací tabulky na vývojový diagram, který obsahuje 7 rozhodovacích podmínek.



Obr. 12 – Transformace tabulky na rozhodovací diagram [15]

Sestupným seřazením podmínek podle počtu definovaných pravidel lze získat menší počet rozhodovacích podmínek, [16] viz Obr. 13, kde před transformací na rozhodovací diagram bylo zaměněno pořadí podmínek *P2* a *P3*. Tím byl redukován počet rozhodovacích podmínek ze 7 na 5.



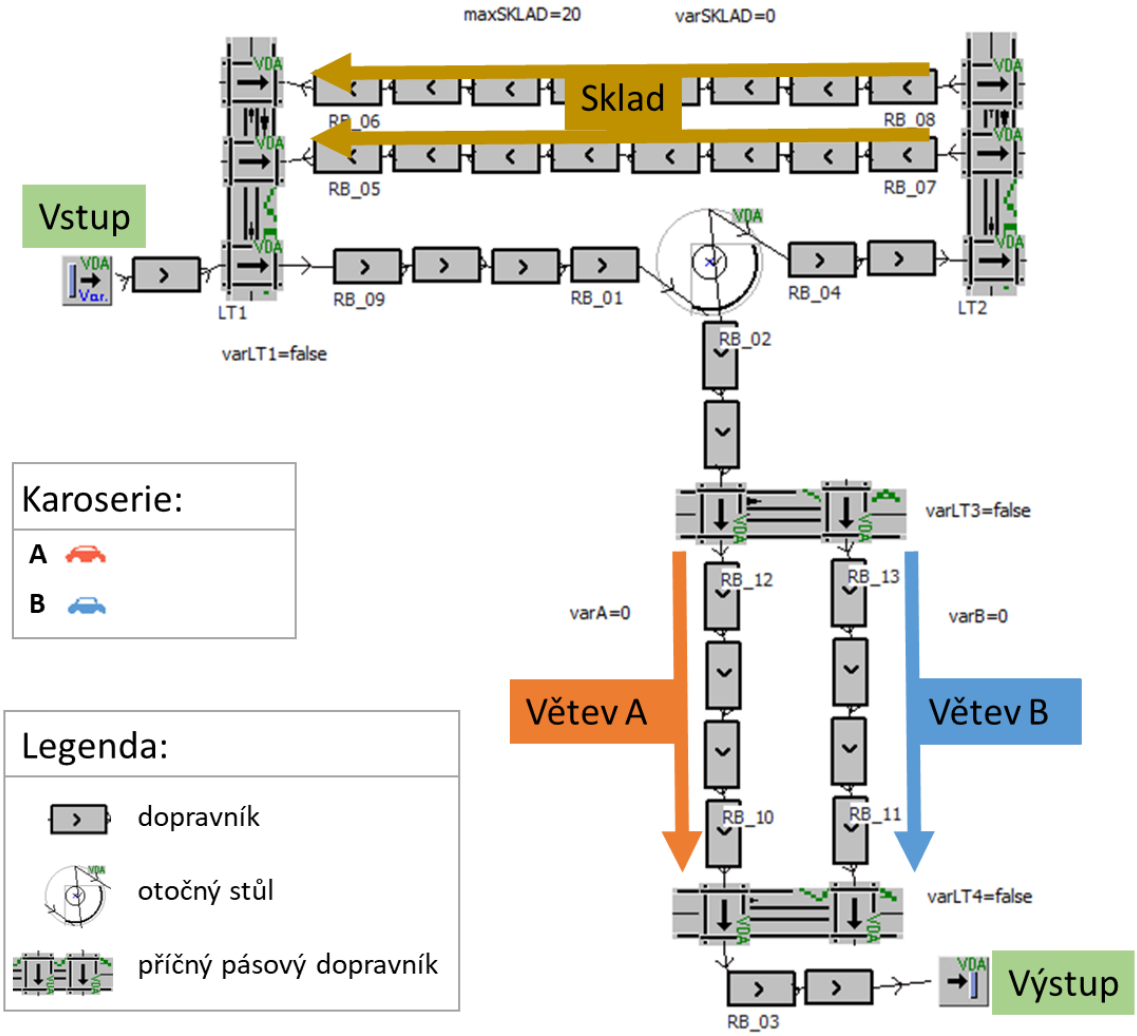
Obr. 13 – Transformace seřazené tabulky na rozhodovací diagram[16]

Seřazení tabulky je vhodné pouze v aplikacích, kde změna pořadí podmínek nezmění vyhodnocení daného problému nebo pokud platnost podmínek neovlivňuje existenci či smysluplnost podmínek dalších. Například algoritmus pro určení barvy výrobku na dopravníku nejprve vyhodnotí existenci výrobku (P2) na dopravníku a následně barvu výrobku (P3). Je nelogické určovat barvu výrobku, pokud se na dopravníku žádný výrobek nenachází.

Praktická část

5. Popis zkoumané oblasti

Pro účely DP byl vytvořen testovací simulační model (Obr. 14), který není obrazem reálného systému. Jedná se o simulační model, na kterém budou demonstrovány a porovnány dva přístupy k tvorbě řídicí logiky pomocí stávajícího programového kódu a navržené rozhodovací tabulky. Systém byl řádně validován a verifikován. Předpokladem pro korektní porovnání stávající a navržené řídicí logiky je stejná průměrná průchodnost simulačního modelu.



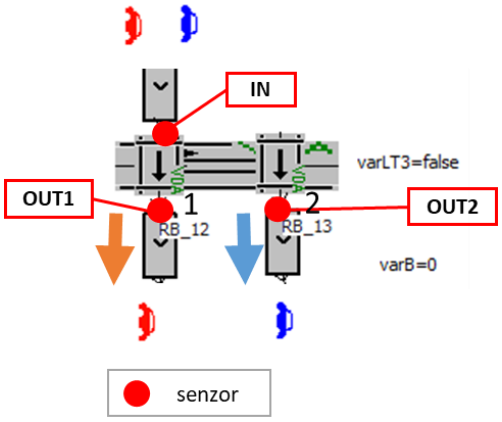
Obr. 14 – Simulační model

V levé horní části je označen vstup a v pravé spodní části výstup ze sledované oblasti. Směr materiálového toku je patrný z orientace šipek dopravníků. Do systému vstupují karoserie typu A a B, které jsou pomocí otočného stolu, dopravníků a příčných pásových dopravníků přesouvány do větve A nebo B dle typu karoserie. Při zaplnění dané větve jsou karoserie přepraveny do skladu, který obsahuje horní a spodní dopravníkovou větev. Primárně je zaplňována spodní dopravníková větev a následně horní. Karoserie ze skladu jsou přesouvány příčným pásovým dopravníkem dolů mezi karoserie, které do modelu vstupují. Pro popsanou funkčnost je nutné definovat logiku řízení jednotlivých prvků simulačního modelu.

5.1 Logika řízení simulačního modelu

Logika řízení je ovlivňována senzory (impulsy) jednotlivých prvků, se kterými lze dále pracovat v rámci řídicí logiky. Dále bude popsána logika řízení příčného pásového dopravníku LT3, který přesouvá karoserie do větve A nebo B dle jejich typu (Obr. 15). V rámci řídicí logiky je pracováno se třemi senzory:

- *IN* – požadavek na přesun karoserie pomocí příčného pásového dopravníku
- *OUT1, OUT2* – karoserie opustila příčný pásový dopravník



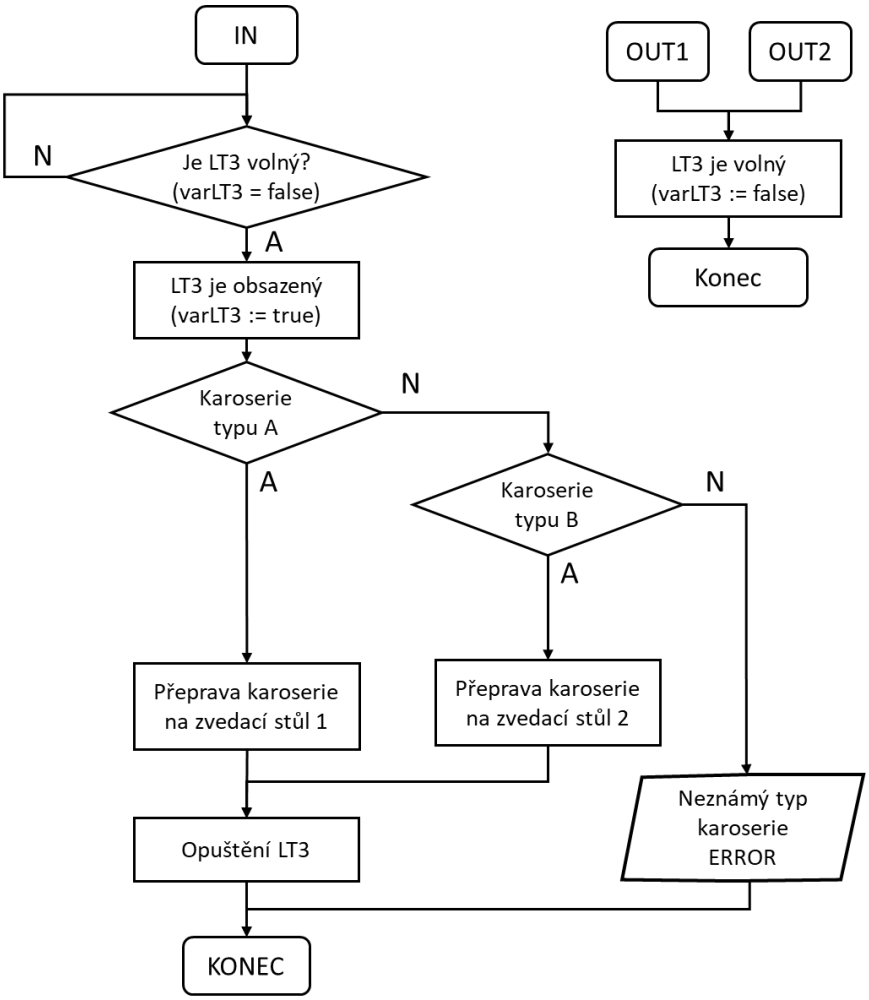
Senzory	
Název	**Prvek**
IN	Příčný pásový dopravník LT3 (vstup)
OUT1	Dopravník RB_12 (vstup)
OUT2	Dopravník RB_13 (vstup)

Obr. 15 – Senzory řízené oblasti

Pro určení obsazenosti stolu slouží *boolean* proměnná *varLT3*, která nabývá hodnot:

- *False* – stůl je volný
- *True* – stůl je obsazen

Následující vývojový diagram (Obr. 16) popisuje logiku řízení příčného pásového dopravníku *LT3* při využití zvolených impulsů.



Obr. 16 – Vývojový diagram logiky řízení

Obdobným způsobem jsou definovány impulsy a řídicí logika i ostatních prvků simulačního modelu. Logiky řízení jednotlivých prvků jsou se zvolenými senzory a vývojovými diagramy uvedeny v příloze A.

Vývojové diagramy slouží pro snadnou čitelnost logiky řízení, avšak jediná možnost implementace řídicí logiky do prostředí PS je pomocí programového kódu.

5.1.1 Řízení programovým kódem

Programovým kódem lze zapisovat několika způsoby. Mezi nejzákladnější patří:

- Přímo do metod jednotlivých prvků (interních metod), které jsou zpracovány při detekci MU na senzoru
- Pomocí prvku *Method* (metody), kterou je nutno propojit se senzorem (*Controls*) daného prvku

V rámci řízení celého simulačního modelu jsou využívány pouze interní metody a následně bude zobrazen programový kód pro řízení příčného pásového dopravníku LT3, který byl popsán v kapitole 5.1.

Metoda příčně přejezdového stolu LT3

```
if ~.varLT3 = false
    -- stůl je volný, karoserie
    -- jede do příslušné větve
    ~.varLT3 := true

    switch p_MU.Variante1
    case "A"
        p_LineDest  := 1
        p_TableDest := "Succ"

    case "B"
        p_LineDest  := 2
        p_TableDest := "Succ"

    end

else
    -- karoserie čeká na vstupu
    p_LineDest  := p_LinePos;
    p_TableDest := p_TablePos;

end
```

Metoda dopravníku RB_12

```
-- stůl je volný
varLT3 := false;
```

Metod dopravníku RB_13

```
-- stůl je volný
varLT3 := false;
```

Obr. 17 – Programový kód v interních metodách

Logiku řízení lze definovat pomocí jedné metody, avšak u komplexních řídicích logik je běžně využíváno více metod pro definování jedné řídicí logiky. Uvedený příklad lze definovat pomocí jedné interní metody, ale pro ukázku byla logika řízení definována pomocí tří interních metod.

Obdobným způsobem je definována řídicí logiky i ostatních prvků simulačního modelu. U řídicích logik definovaných tímto způsobem nemusí být zřejmé, které prvky simulačního modelu nebo části programového kódu v jednotlivých metodách spolu souvisí. To vede ke složité dekompozici logiky řízení.

6. Vývoj a aplikace rozhodovací tabulky

Rozhodovací tabulka označována dále z německého *Entscheidungs-tabelle* jako ET umožňuje přehledně definovat řídicí logiku formou tabulky v PS.

ET slouží jako rozhraní mezi uživatelem a programovým kódem. Umožňuje vytvářet a editovat řídicí logiku v přehledné tabulkové formě, ale v konečném důsledku je simulační model řízen programovým kódem. Konkrétně metodou, která je vygenerována na základě nadefinované řídicí logiky pomocí ET.

Tvorba programového kódu z rozhodovacích tabulek je běžně používaná pro převod z přehledné podoby algoritmů do strojových příkazů počítače. Pro využití v SW PS může být vhodná z hlediska výpočetního výkonu při simulačním běhu a detekci chyb při definování logiky řízení.

- Při zpracování programového kódu, který se skládá z rozhodovacích podmínek *if – else* je výpočetní výkon méně zatížen než při zpracování *for* cyklů pro lokalizaci podmínek v tabulce. Programový kód z rozhodovací tabulky bude vygenerován před spuštěním simulačního běhu a následně bude simulační model řízen pouze tímto kódem.
- Při tvorbě programového kódu je kompilátorem kontrolována existence objektů, správnost datových typů, atd. Vygenerováním programového kódu z rozhodovací tabulky budou kontrolní funkce kompilátoru zachovány. Je nevhodné pokoušet se o náhradu všech funkcí kompilátoru z několika důvodů:
    - Programováním rozsáhlých a komplexních funkcí kompilátoru by se pouze vytvořila kopie již existující věci.
    - S ohledem na rozsáhlost a komplexnost kompilátoru, by nebylo dosaženo totožné funkce v jejím plném rozsahu.
    - SW je neustále aktualizován firmou Siemens. Každá aktualizace kompilátoru by musela být zohledněna do ET.

6.1 Koncept

Návrh tabulky vychází z obecné struktury rozhodovacích tabulek. Stejně jako u obecných rozhodovacích tabulek bude využíváno podmínek a akcí (opatření). Ostatní oblasti jsou navrženy vzhledem k potřebám řízení simulačních modelů v PS. Navržená tabulka je rozdělena do pěti částí (viz Obr. 18) klíčovými slovy: *parametrizace, IMPULS, PODMÍNKA, LOKÁLNÍ PROMĚNNÁ, GLOBÁLNÍ PROMĚNNÁ* a *OPATŘENÍ.*

	string 0	str 1	string 2	strin 3	strin 4	strin 5	strin 6
1	parametrizace		popis				
2	p_MU:object, impuls:string -> string						
3	IMPULS						
4							
5							
6	PODMÍNKA						
7							
8							
9	LOKÁLNÍ PROMĚNNÁ						
10							
11	GLOBÁLNÍ PROMĚNNÁ						
12							
13	OPATŘENÍ						
14							
15							

Obr. 18 – Koncept ET

Struktura tabulky:

- Sloupec č. 0 – definuje parametrizaci, impulsy, podmínky, proměnné a opatření, se kterými bude v rámci tabulky pracováno
- Sloupec č. 1 – slouží k uživatelským funkcím přidání řádku, otevření metody, atd. (více v kapitole 6.4.2)
- Sloupec č. 2 – pro zápis uživatelských poznámek
- Sloupce >= 3 – slouží k definování řídicí logiky

Parametrizace definuje vstupní parametry jednotlivých impulsů, se kterými bude uživatel pracovat (více v kapitole 6.2.2).

Oblast *IMPULS* představuje senzory (místa v simulačním modelu), odkud je ET volána. Při přejezdu MU přes senzor je volána řídicí tabulka a na základě podmínek je zvoleno příslušné opatření.

Do oblasti *PODMÍNKA* jsou vypisovány podmínky podobně jako příkazem *if*. K definování podmínky jsou používány logické operátory =, <, >, atd. Podmínky jsou vyhodnocovány dvoustavovou logikou:

- ANO – *A (případně Y, T nebo 1)*
- NE – *N (případně F nebo 0)*.

LOKÁLNÍ PROMĚNNÁ slouží pro definici lokálních proměnných (dále LP). Zápis LP je obdobný jako při definování LP programovým kódem: „*název LP*: *datový typ* := *hodnota*“.

GLOBÁLNÍ PROMĚNNÁ slouží pro definici globálních proměnných (dále GP). Umožňuje dvě formy zápisu:

- „*název GP*: *datový typ* := *hodnota*“ – vytvoření nové proměnné
- „*název GP*“ – již existující proměnná v modelu, se kterou bude v rámci ET pracováno

OPATŘENÍ určuje příkazy, které se vykonají při splnění daných podmínek. Je možné zde uplatnit jednoduché příkazy, např. přiřazení hodnot do proměnných, ale i metody, které budou obsluhovat komplexní procedury.

Při nevyužití některé oblasti tabulka umožňuje smazání klíčového slova, a tím redukci tabulky do přehlednější formy.

6.1.1 Definice řídicí logiky

Pro popis definice řídicí logiky pomocí ET byla předdefinována logika řízení (Obr. 19), kde není pracováno s GP.

	string 0	st 1	string 2	stri 3	stri 4	stri 5	stri 6
1	parametrizace		popis				
2	p_MU:object, impuls:string -> string						
3	IMPULS						
4	Exit EH5021		Výjezd z otoč...			x	x
5	Enter RB05		Vjezd na dopr...	x	x		
6							
7	PODMÍNKA						
8	RB05.Occupied = false			A	A		
9	p_MU.Variante1 = "A"			A	N	A	N
10	p_MU.Variante1 = "B"			N	A	N	A
11							
12							
13	LOKÁLNÍ PROMĚNNÁ						
14	i : integer			1	2	1	2
15							
16	OPATŘENÍ						
17	promA := promA + i			x	x		
18	promB := promB + i			x	x		
19	promA := promA - i					x	x
20	promB := promB - i					x	x

Obr. 19 – Ukázka řídicí logiky v ET

V oblasti *IMPULS* pomocí písmene *„x“* jsou označeny sloupce, které definují logiku řízení pro daný impuls. Impulsu *Exit EH5021* (ř. 4) je přiřazena logika ve sloupcích 5 a 6.

PODMÍNKA vyhodnocuje splnění či nesplnění podmínek pomocí *„A“ (případně Y, T nebo 1)* a *„N“ (případně F nebo 0)*. (ř. 8 až 10).

Do oblasti *LP* či *GP* se vypisují hodnoty, kterých bude příslušná proměnná nabývat při splnění podmínek daného sloupce (viz ř. 14, kde proměnné *i* jsou přiřazeny hodnoty 1 nebo 2).

OPATŘENÍ jsou definována značkou *„x“* a označují řádek, který je vykonán při splnění podmínek daného sloupce (ř. 17 až 20).

6.2 Implementace do softwaru Plant Simulation

Uživatel bude pro definici řídicí logiky využívat převážně hlavní definiční tabulku, jejíž struktura byla popsána v kapitole 6.1. Ke správné funkčnosti a provázanosti se simulačním modelem tabulka využívá další pomocné tabulky a proměnné. Pro zaručení uživatelské přívětivosti se pomocné tabulky a proměnné vyplňují automaticky s ohledem na aplikační výjimky, kdy bude zapotřebí provázat prvky simulačního modelu s tabulkou ručně.

Dále bude pojmem ET označován parametrizační prvek se všemi jeho funkcemi tabulky a proměnnými. Hlavní definiční tabulkou bude označována tabulka pro definování řídicí logiky.

6.2.1 Impulsy

Řádek impulsu v hlavní definiční tabulce definuje, se kterým prvkem simulačního modelu řídicí logika pracuje. K provázání impulsu a prvku složí tabulka *t_Impuls* (viz pravá část Obr. 20). Jedná se o pomocnou tabulku se třemi sloupci:

Hlavní definiční tabulka

	string 0	st 1
1	parametrizace	
2	p_MU:object, impuls:string -> string	
3	IMPULS	
4	dopravnikA(Entrance_Front)	
5	dopravnikB(Entrance_Front)	
6	dopravnikC(Entrance_Front)	
7	staniceA(Entrance)	
8		
9		
10	PODMÍNKA	

Tabulka *t_Impuls*

string 0	string 1	string 2
impuls	metoda	MU
dopravnikA(Entrance_Front)	~.dopravnikA.Entrance_Front	@
dopravnikB(Entrance_Front)	~.dopravnikB.Entrance_Front	@
dopravnikC(Entrance_Front)	~.dopravnikC.Entrance_Front	@
staniceA(Entrance)	~.staniceA.Entrance	@

Obr. 20 – Provázání hlavní definiční tabulky a tabulky t_Impuls

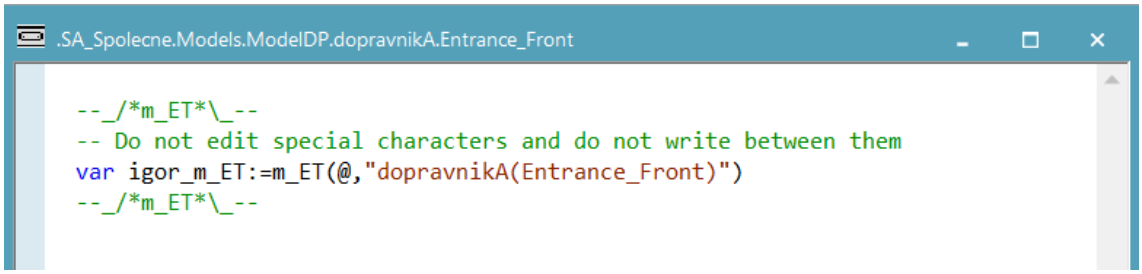
- 0. sloupec definuje názvy impulsů, které se svými jmény a pořadím shodují s impulsy v hlavní definiční tabulce. Názvy a pořadí jsou automaticky propisovány a seřazovány podle hlavní definiční tabulky.
- 1. sloupec definuje prvky simulačního modelu (respektive metody senzoru daných prvků).

- 2. sloupec definuje MU, tedy objekt nacházející se na daném prvku. U prvků standardu VDA je identifikátor pro objekt *p_MU* a pro prvky PS je identifikátorem @, proto je nutno mít možnost definovat různé identifikátory pro jednotlivé impulsy.

Definování tabulky *t_Impuls* je převážně řešeno automaticky pomocí uživatelského dialogu pro zápis impulsů (více v kapitole 6.4.1). Tím je uživateli usnadněno provázání impulsů a simulačního modelu, ale zároveň je tabulka *t_Impuls* editovatelná pro ruční definici nestandardních případů, kdy prvkem provázaným s tabulkou může být například metoda (*Method*), která nemá žádné senzory ani identifikátory objektů.

Tímto je provázána tabulka s jednotlivými impulsy, zbývá provázat jednotlivé impulsy s tabulkou. Do metody daného prvku je potřeba doplnit syntaxi pro volání řídicí tabulky, respektive následně vygenerované řídicí metody. Tato syntaxe bude dále označována jako volací řádek. Ten se může měnit v závislosti na parametrizaci tabulky, proto je v metodě impulsu ohraničen speciálními znaky --/**m_ET***\--, které určují oblast zápisu. Při využití uživatelského dialogu pro zápis impulsů (viz kapitola 6.4.1) jsou speciální znaky do metod přidány automaticky. Pro zachování univerzálnosti může uživatel oblast vymezenou speciálními znaky přemístit ve struktuře metody daného prvku výše či níže.

Následující Obr. 21 ukazuje zápis volací syntaxe do interní metody *Entrance_Front* prvku *dopravnikA* (viz ř. 4 na Obr. 20).



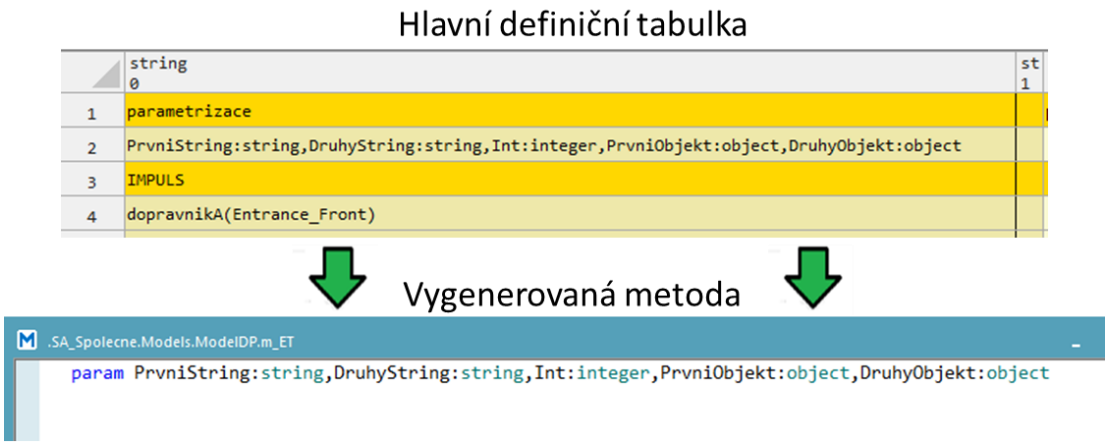
Obr. 21 – Zápis volací syntaxe

6.2.2 Parametrizace řídicí metody

Zapisuje se v horní části tabulky pod klíčové slovo *parametrizace* obdobně jako v programovacím jazyce SimTalk: *název1:datový typ1, název2:datový typ2, ...* → *návratový datový typ*. Parametrizace je předvyplněná jako *p_MU:object, impuls:string* →

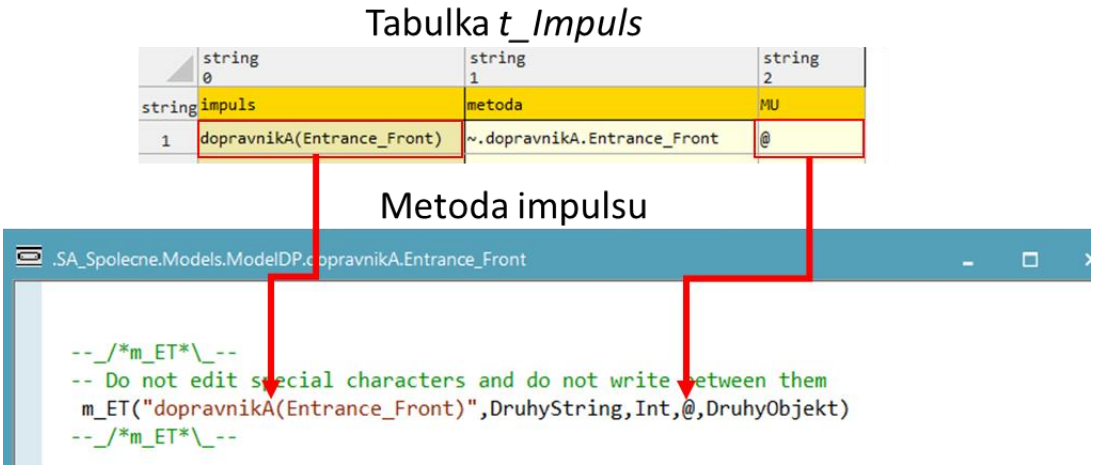
string, ale uživatel ji může přepsat do libovolné podoby. Je však nutné myslet na význam jednotlivých parametrů.

Parametrizaci v hlavní definiční tabulce odpovídá parametrizace vygenerované metody. Pro upravenou parametrizaci na ř. 2 hlavní definiční tabulky Obr. 22 by vygenerovaná metoda vypadala následovně.



Obr. 22 – Upravené parametrizace – vygenerovaná metoda

Pro tuto parametrizaci by impulsu *dopravnikA(Entrance_Front)* odpovídal následující tvar volacího řádku (Obr. 23).



Obr. 23 – Upravená parametrizace – volací řádek

První parametry datových typů *string* a *object* jsou ve volacím řádku vyplňovány na základě tabulky *t_Impuls*:

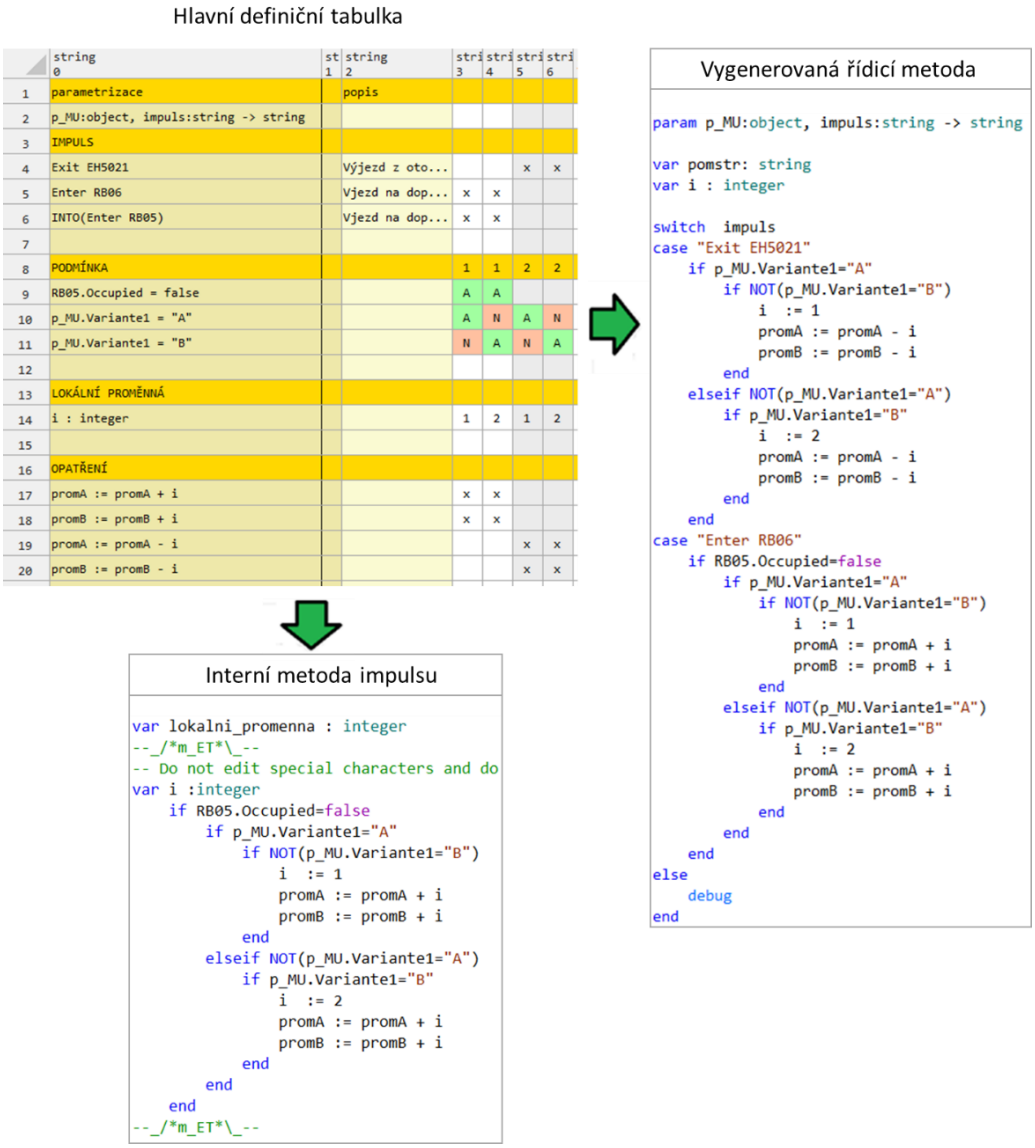
- *string* je název impulsu (sloupec *impuls*)

- *object* je identifikátor objektu na daném prvku (sloupec *MU*)

Parametrizace je uživatelem nastavována pomocí jedné buňky v hlavní definiční tabulce a struktura parametrizace se zrcadlí do všech volacích řádků automaticky. Pokud je definována návratová hodnota, tak je v metodě impulsu automaticky vytvořena lokální proměnná pro uložení návratové hodnoty.

6.2.3 Generování programového kódu

Z definiční tabulky bude automaticky vygenerována řídicí metoda. Transformace z hlavní definiční tabulky na řídicí metodu je uvedena na Obr. 24.



Obr. 24 – Transformace ET na programový kód

Vyplněná definiční tabulka je zpracována transformačním algoritmem, který řeší vnoření a návaznost podmínek a vygeneruje řídicí metodu s programovým kódem. Při generování jsou aktualizovány veškeré volací řádky v metodách impulsů a simulační model je ihned připravený k testování nadefinované řídicí logiky (i rozsáhlejší tabulky přibližně nad 60 řádků, jsou transformovány v řádech desetin sekund).

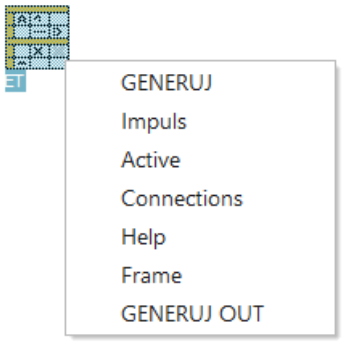
Vygenerovaná metoda je strukturována příkazem *switch-case*. První předaný parametr typu *string* je název impulsu pro jednotlivé *case*. Při volání řídicí metody z několika impulsů je zpracován předaný parametr *string* a vykonají se programové řádky pouze pro daný impuls (viz vygenerovaný kód v pravé části Obr. 24).

Z důvodu kompatibility s různými případy řízení může být logika generována přímo do interních metod impulsů mezi speciální znaky namísto volacího řádku. Například prvky standardu VDA mají velké množství lokálních proměnných pro své interní řízení, a proto by muselo být předáváno i vraceno velké množství parametrů. Pokud kód bude vepsán přímo do metody takového prvku, pak lze rovnou s lokálními proměnnými pracovat. Impulsy, u nichž má být kód generován do interní metody, se označují klíčovým slovem *INTO* před názvem impulsu. Toto je zobrazeno na Obr. 24 v hlavní definiční tabulce na 6. řádku a interní metoda s vygenerovaným kódem se nachází ve spodní části tohoto obrázku.

Struktura podmínek vygenerovaného programového kódu nebude optimální, protože podmínky budou generovány strojově. Generovaný kód lze optimalizovat transformačním algoritmem. Vývoj tohoto algoritmu je dalším krokem projektu. Jeho zpracování není v časových možnostech této DP, neboť cílem je vyvinout rozhraní pro přehlednější tvorbu logik řízení. Optimalizaci nelze provést jednoduchou formou záměny podmínkových řádků (předzpracováním tabulky), neboť by tím došlo k narušení chronologie podmínek, což by pro simulační modely mohlo vést k nefunkčnosti řídicí logiky. Bylo by nutné vyvinout sofistikovanější způsob generování kódu, což by momentálně zastínilo vývoj ostatních funkcí rozhodovací tabulky. Dosavadní nedostatek je kompenzován přehledností řídicí logiky v hlavní definiční tabulce, se kterou bude pracováno primárně.

6.3 Kontextové menu a generování metod

Dvojklikem levého tlačítka myši na prvek ET se otevře hlavní definiční tabulka. Naopak klikem pravého tlačítka je rozbalena uživatelská nabídka, viz Obr. 25.



Obr. 25 – Uživatelská nabídka ET

První volbou *GENERUJ* je vytvořena metoda s názvem *m_ET*, která je určena primárně pro testování nadefinované logiky, dále bude označována jako testovací metoda. Aktuálně rozpracovaná logika v ET je zachována, uživatel může testovat a upravovat řídicí logiku.

Poslední volbou *GENERUJ OUT* je uživatel vyzván k zadání názvu metody, která je následně vygenerována. Tato metoda bude označována jako exportovaná metoda a na rozdíl od testovací metody vymaže dosavadně definovanou logiku v ET. Tím je tabulka uvolněna pro definování další řídicí logiky. Exportované metodě jsou přiřazeny atributy pro pozdější možnost editace a jsou přepsány veškeré volací řádky v impulsech, aby korespondovaly s názvem exportované metody (více v kapitole 6.3.1).

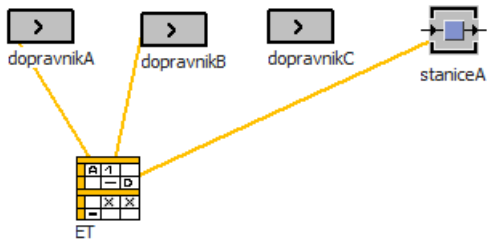
Impuls otevře tabulku *t_Impuls*, kde uživatel může editovat provázání impulsů s modelem dle kapitoly 6.2.1.

Active zobrazí tabulku všech exportovaných metod (Obr. 26), kde pomocí sloupce *active* lze aktivovat či deaktivovat jednotlivé metody. Deaktivací metody se před všechny řádky programového kódu i volací řádky jednotlivých impulsů přidá znak „--“, což z nich udělá komentář. Naopak aktivace metody znaky komentáře odebere. Lze tak provádět experimenty na jediném simulačním modelu s několika scénáři řízení, což je bez využití prvku ET velmi obtížné.

	object 0	boolean 1
string	metoda	active
1	*.SA_Spolecne.Models.ModelET.m_rizeni2	true
2	*.SA_Spolecne.Models.ModelET.m_rizeni	false
3	*.SA_Spolecne.Models.ModelET.m_rizeni1	true
4	*.SA_Spolecne.Models.ModelET.m_rizeni3	true

Obr. 26 – Aktivace, deaktivace metod

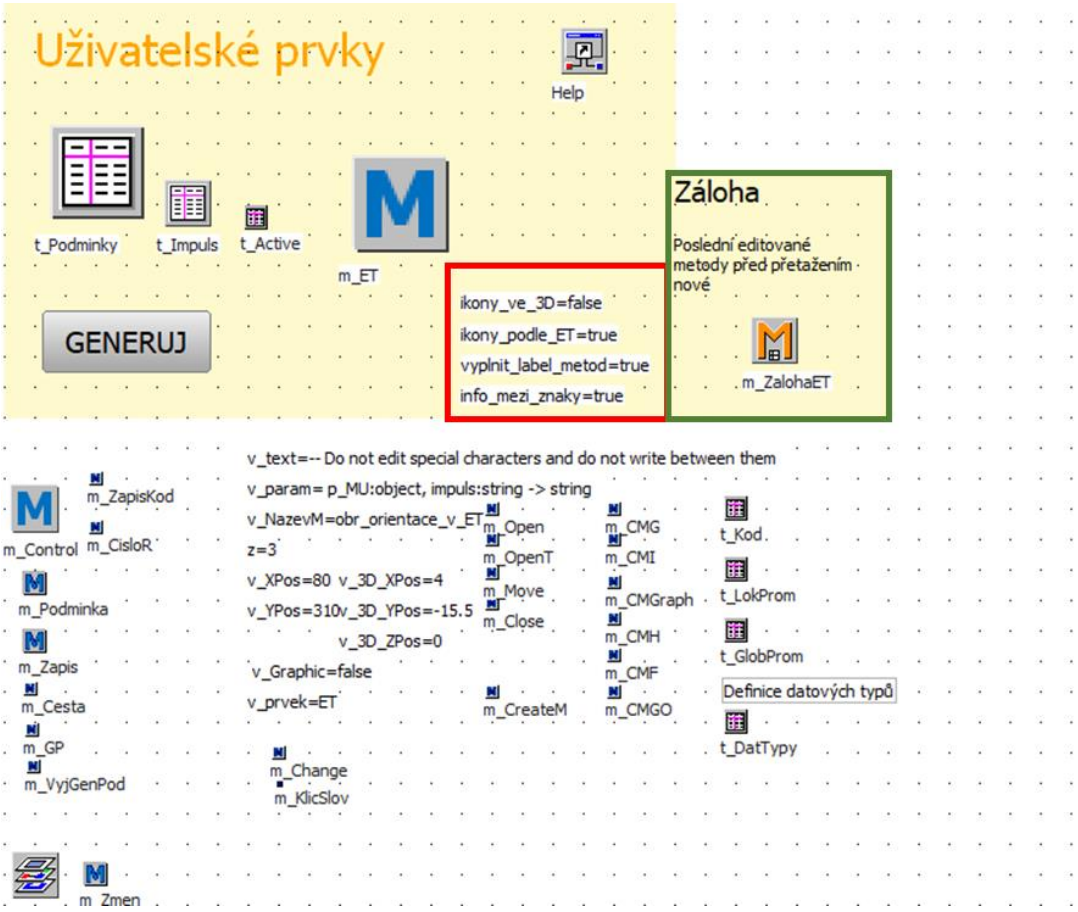
Connections vytvoří spojnice mezi prvkem ET a prvky jednotlivých impulsů, což slouží pro rychlou orientaci mezi prvky ovlivňující definovanou logiku řízení, viz Obr. 27, kde ET pracuje s prvky *dopravnikA, dopravnikB* a *staniceA*.



Obr. 27 – Spojnice ET a simulačních prvků

Help otevře prezentaci se stručnými návody a příklady definování ET.

Frame zobrazí podsoubor prvku ET, kde jsou metody, tabulky a proměnné obsluhující funkce prvku. Pro uživatele je užitečná žlutě podbarvená oblast na Obr. 28, kde v červeném rámečku jsou označené proměnné pro nastavení parametrů generovaných metod.



Obr. 28 – Podsoubor prvku ET

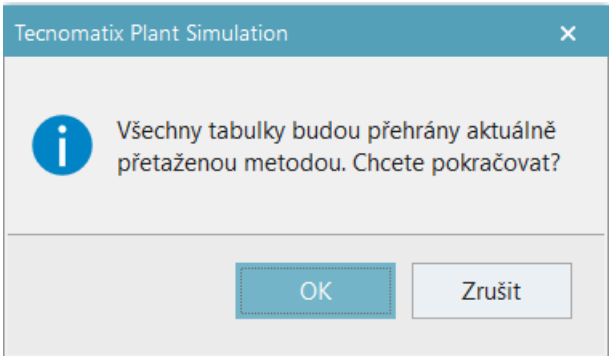
Popis proměnných pro parametry vygenerovaných metod:

- *ikony_ve_3D* – slouží ke generování metody na správné pozici pod prvkem ET, protože rastr okna sítě ve 2D a 3D není totožný
- *ikony_podle_ET* – určuje, zdali vygenerovaná metoda bude mít stejnou velikost jako prvek ET
- *vyplnit_label_metody* – vyplnit, nevyplnit popisek generované metody
- *info_mezi_znaky* – propsat, nepropsat řetězec *Do not edit special charakters and do not write between them* mezi speciální znaky *--/*m_ET*\--* v jednotlivých metodách impulsů

6.3.1 Editování metod

Exportované metody jsou editovatelné přetažením na prvek ET. Aby však nedošlo k přepsání aktuálně definované logiky v ET, je zvoleno několik opatření na základě stavu rozpracovanosti ET:

- V ET není aktuálně rozpracovaná žádná řídicí logika:
  Přetažením exportované metody se automaticky nahrají veškeré tabulky a prvky přetažené metody.
- V ET je rozpracovaná řídicí logika, která již v minulosti byla exportovaná:
  Aktuální logika v ET se vygeneruje jako exportovaná metoda pod názvem, jenž uživatel v minulosti zadal a následně se přetažená metoda nahraje do ET.
- V ET je rozpracována řídicí logika, která ještě nebyla exportovaná:
  Uživatel je upozorněn na přepsání logiky v ET (viz Obr. 29). Pokud aktuálně rozepsaná logika nemá být vymazána, uživatel by měl zvolit *Zrušit*, logiku exportovat a následně přetáhnout editovanou metodu. Kdyby uživatel omylem zvolil *OK*, je přemazaná metoda dohledatelná v prvku ET (v zeleném obdélníku Obr. 28), kde se metoda vygeneruje jakožto m*_ZalohaET*, avšak lze dohledat pouze naposledy přemazanou metodu.



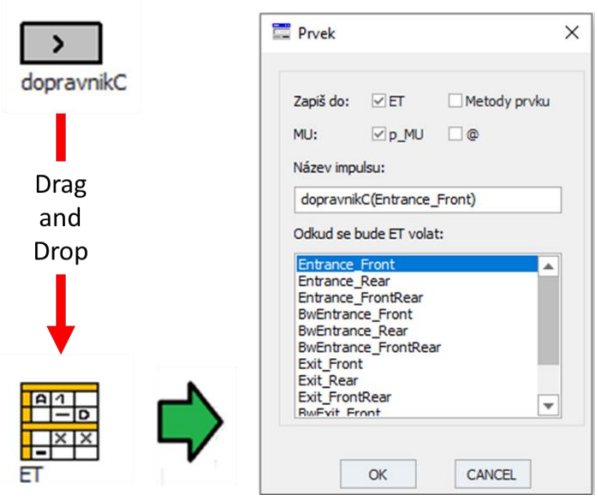
Obr. 29 – Upozornění při editování

6.4 Uživatelské rozhraní

Pro snadnější provázání ET se simulačním modelem vzniklo uživatelské rozhraní pro zápis impulsů, generování metod, nápovědu při psaní podmínek, grafické zvýraznění podmínek a klíčových slov, zkratky pro zápis klíčových slov, atd.

6.4.1 Definice impulsů

Pro definici impulsů bylo vytvořeno drag and drop („chyť a pust“) rozhraní. Uživatel přetáhne požadovaný prvek na ET a následně je otevřen dialog pro definici impulsu (viz Obr. 30).



Obr. 30 – Dialog pro zápis impulsů

V horní části dialogu jsou zaškrtávací pole pro určení místa zápisu kódu. Algoritmus obsluhující uživatelský dialog nepovolí výběr obou polí najednou, ani nevybrat pole žádné.

- *ET* – zápis do vygenerované řídicí metody
- *Metody prvku* – zápis do interní metody prvku

Dále je volen identifikátor MU. Nabízeny jsou dva, ale po ukončení definice impulsu lze v tabulce *t_Impuls* libovolně identifikátor měnit.

- *p_MU* – pro prvky standardu VDA
- *@* – pro základní prvky PS

V textovém poli *Název impulsu* je řetězec, pod kterým bude impuls zapsán do hlavní definiční tabulky. Nabývá tvaru *název prvku (metoda)*, ale lze jej přepsat do libovolné podoby. Ve spodní část dialogu je výběrové pole dostupných senzorů přetaženého prvku.

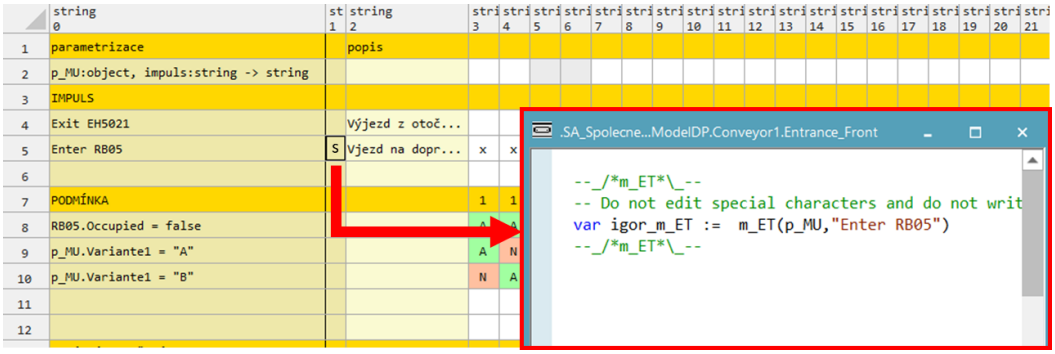
Po potvrzení dialogu je na přetaženém prvku založena interní metoda pro vybraný senzor, je automaticky propsána do *Controls* přetaženého prvku a do metody je zapsán volací kód. Impuls se automaticky propíše do hlavní definiční tabulky a pomocné tabulky *t_Impuls* s příslušným odkazem na prvek a identifikátorem MU.

Při odstranění impulsu z hlavní definiční tabulky je prvek provázaný s impulsem uveden do původního stavu. Automaticky jsou z metody odstraněného impulsu odebrány veškeré znaky přidané prvkem ET a pokud v metodě nezbyl žádný programový kód, je uvolněna z *Controls* daného prvku a odstraněna.

6.4.2 Pomocné uživatelské funkce

Pro pomocné funkce je vyhrazený 1. sloupec hlavní definiční tabulky. Zapsáním číslice je přidán požadovaný počet prázdných řádků pod aktivní řádek, protože editor tabulek pro PS umožňuje přidání pouze 1 řádku. Naopak zapsáním libovolného znaku mimo číslice je otevřena metoda impulsu nebo pomocná metoda.

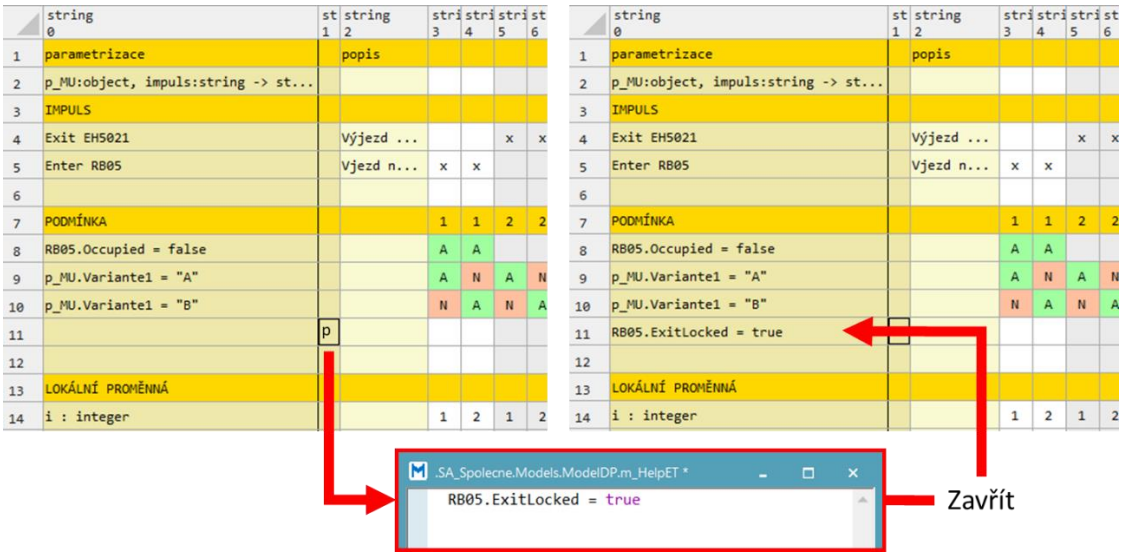
- Metoda impulsu je otevřena zápisem libovolného znaku do oblasti *IMPULS* (viz Obr. 31). Uživatel může velmi rychle otevřít a editovat libovolnou metodu impulsu.



Obr. 31 – Otevření metody impulsu

- Pomocná metoda je otevřena zápisem libovolného znaku do jiné oblasti než *IMPULS*. Uživatel může do pomocné metody psát programový kód a tím využít

nápovědu SimTalk kompilátoru. Při zavření metody je napsaný kód propsán do buňky, ze které byla metoda otevřena (viz Obr. 32).



Obr. 32 – Využití pomocné metody

6.4.3 Klíčová slova a grafické znázornění

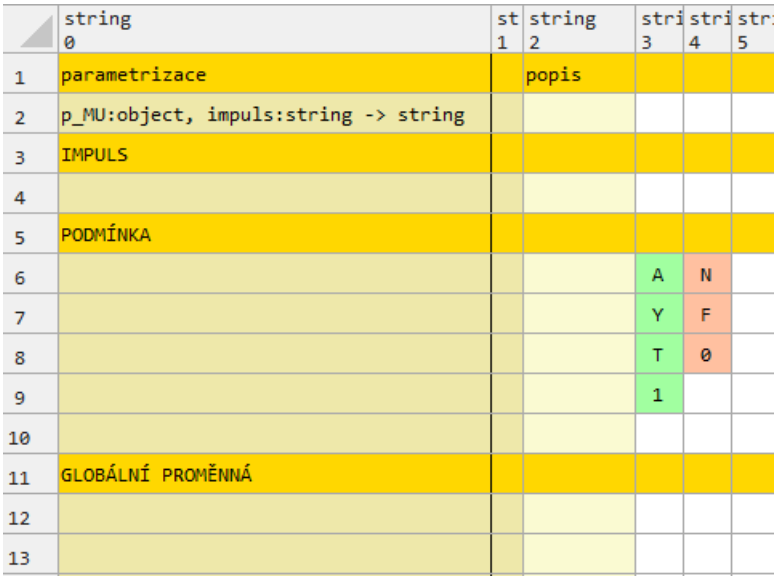
Byl vytvořen algoritmus pro zápis a kontrolu pořadí klíčových slov, která lze odebírat a přidávat dle potřeb logiky řízení. První tři řádky (*parametrizace* až *IMPULS*) nelze odebrat, protože jsou nezbytné pro provázání simulačního modelu a ET. U ostatních klíčových slov žádná omezení nejsou.

Odebraná klíčová slova lze přidat vypsáním celého jejich názvu nebo použitím příslušných zkratek, které jsou uvedeny v Tab. 2. Při nedodržení pořadí jsou klíčová slova automaticky srovnána dle pořadí v Tab. 2.

Tab. 2 – Zkratky klíčových slov

Klíčové slovo	Zkratky
PODMÍNKA	po, pod
LOKÁLNÍ PROMĚNNÁ	lp, lo, lok
GLOBÁLNÍ PROMĚNNÁ	gp, gl, glob
OPATŘENÍ	op

Po odebrání nebo přidání klíčového slova je dle aktuální podoby tabulky automaticky změněno barevné pozadí řádků klíčových slov (viz Obr. 33 ř. 1, 3, 5 a 11). Algoritmus také mění barevné pozadí buněk v sekci *PODMÍNKA* dle rozhodovacích znaků *N, 0, F* - červeně a *A, Y, 1, T* – zeleně (viz Obr. 33 ř. 6, 7, 8 a 9).



	string 0	st 1	string 2	stri 3	stri 4	str 5
1	parametrizace		popis			
2	p_MU:object, impuls:string -> string					
3	IMPULS					
4						
5	PODMÍNKA					
6				A	N	
7				Y	F	
8				T	0	
9				1		
10						
11	GLOBÁLNÍ PROMĚNNÁ					
12						
13						

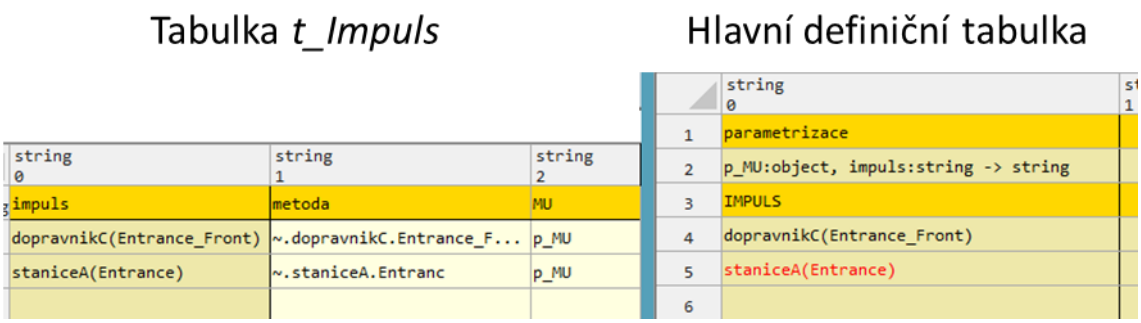
Obr. 33 – Grafické znázornění

6.5 Kontrolní funkce - Watchdog

Je soubor funkcí pro kontrolu správného definování tabulky *t_Impuls*, speciálních znaků v metodách impulsů a hlavní definiční tabulky.

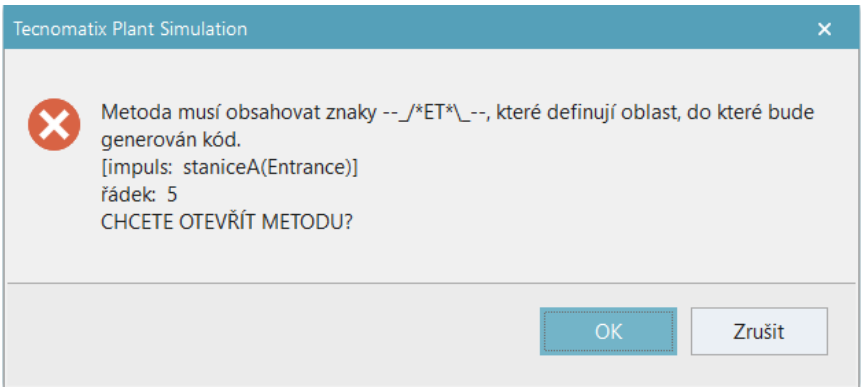
6.5.1 Kontrola impulsů

Při definování impulsů pomocí uživatelského rozhraní je hlavní definiční tabulka vždy správně provázána s tabulkou *t_Impuls*, avšak při ruční definici může být nesprávně definována metoda daného impulsu. V takovém případě název impulsu v hlavní definiční tabulce zčervená, a upozorňuje tím uživatele na nesprávně definovaný objekt (viz 5. řádek hlavní definiční tabulky na Obr. 34).



Obr. 34 – Nesprávná definice impulsu

Při požadavku na vygenerování metody je zkontrolována existence speciálních znaků --/*m_ET*\-- ve všech metodách impulsů. Pokud znaky v metodě chybí, je uživatel upozorněn dialogem (Obr. 35), který obsahuje text chyby, název impulsu ve hranatých závorkách a řádek, na kterém se impuls nachází. Danou metodu lze z dialogu otevřít a znaky doplnit.



Obr. 35 – Chyba zápisu speciálních znaků

6.5.2 Kontrola hlavní definiční tabulky

V některých oblastech hlavní definiční tabulky je kontrolována definice dané oblasti či existence používaných objektů.

V oblasti *parametrizace* je kontrolován správný tvar zápisu. Chyba je indikována pomocí červeného písma.

Při nedodržení správného tvaru zápisu GP nebo LP je proměnná zbarvena červeně. GP může být definována nejenom jako nově vytvořená, ale také jako proměnná simulačního modelu. V takovém případě je kontrolována existence této proměnné.

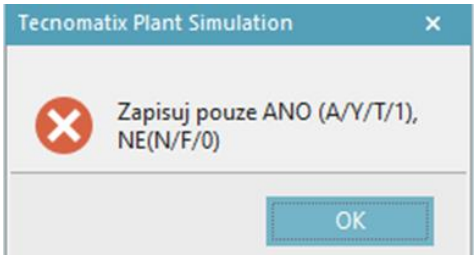
Na následujícím obrázku (Obr. 36) jsou zobrazeny možné chybné zápisy do hlavní definiční tabulky:

	string 0	str 1	string 2	strin 3	strin 4	strin 5	strin 6	s
1	parametrizace		popis					
2	p_MU:object, impuls:string -> strin							
3	IMPULS							
4	Exit EH5021		Výjezd z ...	x	x	x	x	
5								
6	PODMÍNKA			1	1	2	2	
7	RB05.Occupied = false			A	A			
8	p_MU.Variante1 = "A"			A	N	A	N	
9								
10	LOKÁLNÍ PROMĚNNÁ							
11	i - integer			1	2	1	2	
12	retezec : strin							
13								
14	GLOBÁLNÍ PROMĚNNÁ							
15	cislo : integer							
16	promenna_z_modelu							
17	promenna_z_model							
18								
19	OPATŘENÍ							
20	promA := promA + i			x	x			

Obr. 36 – Možné chyby v definici

- Parametrizace vrací nesprávně zapsaný datový typ *strin* (ř. 2)
- Lokální proměnná *i* používá mezi názvem a datovým typem znak „-“ namísto „:“ (ř. 11)
- Proměnné *retezec* je deklarovaný chybný datový typ *strin* (ř. 12)
- Globální proměnná *promenna_z_model* (ř. 17) neexistuje v simulačním modelu

Oblast pro definování splnění či nesplnění podmínek (zelený obdélník na Obr. 36) dovoluje zapsat pouze znaky *A,Y,T,1* nebo *N,F,0*. Jiné znaky jsou automaticky vymazány. Opakovaným zápisem jiných znaků proběhne upozornění uživatele (Obr. 37).



Obr. 37 – Chybný zápis znaků

7. Zhodnocení výsledků

Pro porovnání řízení pomocí programového kódu a rozhodovací tabulky byly vytvořeny dva totožné testovací simulační modely, oba shodné s popsaným simulačním modelem v kapitole 5. Logika řízení simulačních modelů byla vytvořena:

- 1. pomocí programového kódu, viz kapitola 5 (model 1).
- 2. pomocí rozhodovací tabulky viz příloha B. U prvků simulačního modelu jsou exportované metody (prvky s oranžovým *M*) sloužící k jejich řízení. Jednotlivé logiky řízení jsou ve formě tabulek uvedeny v příloze B pod simulačním modelem (model 2).

7.1 Porovnání průměrné průchodnosti modelu

Simulační model 2, byl verifikován a validován vůči simulačnímu modelu 1.

Po celou dobu tvorby modelu 2 byla posuzována logika řízení a struktura modelu, které se shodují s modelem 1. Verifikace je považována za dostatečnou.

Oba simulační modely nezahrnují stochastické vlivy (např. poruchovost), proto mohla být validace modelu 2 ověřena v relativně krátkém simulačním období 10 dnů, což je vzhledem k systémovým hranicím dostatečné. Bylo dosaženo stejné průchodnosti karoserií, proto lze model 2 prohlásit za validovaný (viz Tab. 3).

Tab. 3 – Porovnání průchodnosti

	Vstup		Výstup	
	Typ	Vstupní poměr	Logika řízení vytvořena	
			Programováním	Rozhodovací tabulkou
Karoserie	A	5 : 4	1440 kar.	1440 kar.
	B		1440 kar.	1440 kar.

7.2 Porovnání přístupů k tvorbě řídicí logiky

Chování simulačních modelů z kapitoly 7 je totožné a následně bude porovnána přehlednost řídicích logik a doba trvání simulačního běhu.

7.2.1 Přehlednost řídicí logiky

Při tvorbě logiky řízení programovým kódem byla v tomto případě logika definována pomocí interních metod jednotlivých prvků simulačního modelu. Nemusí být zřejmé, které prvky simulačního modelu danou řídicí logiku ovlivňují. Pokud by řešitel byl postaven před takto naprogramovaný model s neznámým popisem funkčnosti, dával by do souvislosti logiku řízení z jednotlivých interních metod a částí programového kódu, což může být pro dekompozici komplexních logik časově náročné. V případě rozsáhlejších simulačních modelů se sofistikovanější logikou řízení čas dekompozice řídicí logiky narůstá. Problém vzniká i při požadavku na úpravu řídicí logiky. Řešitel by náročně doplňoval další podmínky a akční zásahy tak, aby byly správně začleněné ve funkční struktuře programového kódu a neovlivnily ostatní funkce simulačního modelu.

Při tvorbě řídicí logiky rozhodovací tabulkou lze přehledně určit podmínky příslušící danému zásahu do simulačního modelu. Zejména je čitelné, jaké impulsy (prvky) ovlivňují danou řídicí logiku. Výhodou je úprava řídicí logiky přidáním či odebráním sloupců nebo řádků bez zdlouhavé dekompozice provázanosti jednotlivých podmínek.

7.2.2 Doby trvání simulace

V rámci této kapitoly bude pojmem neefektivní kód myšlen programový kód, jehož podmínky se v rámci kódu mohou několikrát opakovat, a tím vytváří neoptimální rozhodovací strukturu. Opakem neefektivního kódu je kód efektivní.

Pro testovací model byly vytvořeny tři logiky řízení se stejnou funkcí:

- *Řízení bez ET, efektivní kód řízení* – standardní programování interních metod prvků
- *Řízení bez ET, neefektivní kód řízení* – přepsaný kód z exportovaných metod přímo do interních metod prvků simulačního modelu
- *Řízení pomocí ET* – logika řízení vytvořena pomocí prvku ET

V následující tabulce (Tab. 4) jsou uvedeny reálné doby výpočtu simulačních experimentů pro simulované období 10 a 100 dnů pro tři zmíněné logiky řízení se stejnou funkcí.

Tab. 4 – Srovnání časů simulačních běhů

		Simulační čas		
		10 dnů	**100 dnů**	**Zpomalení 10 /100 dnů [%]**
Reálný čas výpočtu	**Řízení bez ET, efektivní kód řízení**	17.8 s	204.4 s	**- / -**
	Řízení bez ET, neefektivní kód řízení	20.3 s	235.8 s	**14 / 15**
	Řízení pomocí ET	20.1 s	233.8 s	**13 / 14**

Neefektivita programového kódu prodlužuje dobu výpočtu simulačního experimentu, avšak řízení simulačních modelů stejně neefektivním kódem pomocí ET nebo bez ET nemá téměř žádný vliv na reálnou dobu výpočtu. Efektivita programového kódu může zkrátit reálný čas výpočtu simulačního experimentu. Při užití odladěného efektivního kódu řízení bylo pro konkrétní testovací příklad dosaženo rychlejší doby výpočtu o maximálně 15 %.

Vytvořit odladěný a efektivní kód je časově náročné, vyžaduje velkou míru zkušeností a snaha o maximální efektivitu kódu může zastínit jiné, někdy i důležitější, požadavky na řízení. Prvek ET sám o sobě zpoždění doby výpočtu do simulačních experimentů nezanáší a efektivitu vygenerovaného kódu lze dále optimalizovat transformačním algoritmem.

8. Závěr

Úvodem byla představena přepravní technologie, která vytváří materiálový tok v průmyslových závodech, k jehož optimalizaci se používají simulační softwary. Byl popsán simulační software Plant Simulation (PS), jeho využití, základní prvky pro tvorbu simulačních modelů a způsob jejich řízení.

Pro vytváření logiky řízení simulačních modelů byl navržen nový způsob řízení pomocí rozhodovací tabulky. S ohledem na potřeby a kompatibilitu se simulačními modely v SW PS byla navržena struktura, popsána orientace v rozhodovací tabulce a způsob zápisu řídicí logiky. Byl vytvořen parametrizační prvek v prostředí PS, pro který bylo vyvinuto rozhraní mezi prvkem a simulačním modelem pomocí generování programového kódu. Ten je generován do řídicích metod na základě definované rozhodovací tabulky. Pro ovládání prvku bylo vytvořeno přehledné uživatelské rozhraní. Z řádně vyplněné rozhodovací tabulky lze generovat řídicí metody „na kliknutí“ a jednoduše provázat jednotlivé prvky simulačního modelu s rozhodovací tabulkou. Vygenerované řídicí metody lze pomocí parametrizačního prvku zpětně dekomponovat i editovat v tabulkové formě.

Práce s rozhodovací tabulkou byla testována na konkrétním simulačním modelu. Pro tento model bylo dosaženo předpokladu totožné funkce při tvorbě řídicí logiky programovým kódem i rozhodovací tabulkou, tudíž bylo možné porovnat výhody a nevýhody těchto přístupů. Tvorba řídicí logiky pomocí programového kódu umožňuje uživateli libovolně strukturovat rozhodovací podmínky. Zkušený programátor může vytvořit optimální programový kód, což přispívá k rychlejšímu výpočtu simulačních experimentů. Při tvorbě řídicí logiky pomocí rozhodovací tabulky je uživatel odkázán na vytvořenou strukturu programového kódu rozhodovací tabulkou. Pro optimalizaci struktury podmínek vygenerovaného programového kódu bude dále, mimo rámec této DP, vyvíjen transformační algoritmus převádějící logiku řízení z rozhodovací tabulky na programový kód. Navržený prvek je využíván v praxi, kde se osvědčil, a je snahou jej postupně rozšířit nejen na simulační modely využívající dopravníkové systémy.

Různé přístupy k tvorbě řídicí logiky pomocí programového kódu vedou k rozmanitosti jednotlivých řešení. Rozhodovací tabulka zavádí standard pro jednotnou tvorbu řídicích logik, což přispívá ke snadnějšímu sdílení projektů. Prvek je a bude průběžně prakticky

testován, případné chyby budou eliminovány a v případě potřeby budou rozvíjeny nové funkce dle budoucích požadavků.

Seznam použité literatury

[1] *SYSTÉMY: Základní pojmy z teorie systémů* [online]. [cit. 2022-02-20]. Dostupné z: http://www.zam.fme.vutbr.cz/~osmera/IS_pdf/D_kap1.pdf

[2] BANKS, Jerry, John S. CARSON, Barry L. NELSON a David M. NICOL. *Discrete-event system simulation: Fourth Edition*. 2005. [cit. 2022-02-18]. ISBN ISBN 978-013-8150-372.

[3] ŠTOČEK, Jiří. *Optimalizace materiálového toku ve vybraném průmyslovém závodě*. Brno, 2004. Dizertační práce. Vysoké učení technické v Brně

[4] VARJAN, Matúš. *Simulační verifikace komplexního technologického projektu*. Brno, 2016. Dizertační práce. Vysoké učení technické v Brně

[5] SEDLÁČEK, Martin. *Projektování logistických řetězců pomocí počítačové simulace*. Brno, 2017. Diplomová práce. Vysoké učení technické v Brně.

[6] MÁČALA, Stanislav. *Ověření dosažení požadované sekvence výrobků při průchodu oblastí lakovny pomocí simulačního modelu*. Brno, 2014. Diplomová práce. Vysoké učení technické v Brně.

[7] HLOSKA, Jiří a Miroslav ŠKOPÁN. *Plánování a řízení provozu pivovaru s podporou počítačové simulace*. Kvasný průmysl [online]. 2014. [cit. 2022-01-05]. Dostupné z: https://www.kvasnyprumysl.cz/pdfs/kpr/2014/06/01.pdf

[8] VOLF, Luděk, Libor BERÁNEK a Petr MIKEŠ. *Počítačová simulace ve strojírenské výrobě* [online]. [cit. 2022-01-06]. ISBN 978-80-7043-934-0. Dostupné z: https://dspace5.zcu.cz/bitstream/11025/16403/1/Volf.pdf

[9] *TECNOMATIX PLANT SIMULATION* [online]. [cit. 2022-01-08]. Dostupné z: https://www.axiomtech.cz/25357-texnomatix-plant-simulation

[10] HLOSKA, Jiří. *Transformace simulačního modelu ze SW SimPro do SW Plant Simulation*. Brno, 2010. Diplomová práce. Vysoké učení technické v Brně.

[11] KŘIVÝ, Luděk, Evžen KINDLER a Petr MIKEŠ. *Simulace a modelování* [online]. [cit. 2022-01-06]. Dostupné z: https://vendulka.zcu.cz/Download/Free/SkriptaKindlerMS.pdf

[12] *POSTUPY, RIEŠENIE PROBLÉMOV, ALGORITMICKÉ MYSLENIE* [online]. [cit. 2022-02-19]. Dostupné z: http://files.inf-gtv.webnode.sk/200000045-bdb5ebeb1d/01-02-ALG-1.pdf

[13] *IBM: Business Process Manager on Cloud* [online]. [cit. 2022-02-08]. Dostupné z: https://www.ibm.com/docs/en/bpmoc?topic=SS964W/com.ibm.wbpm.admin.doc/topics/cbre_busiru_decisiontable.html

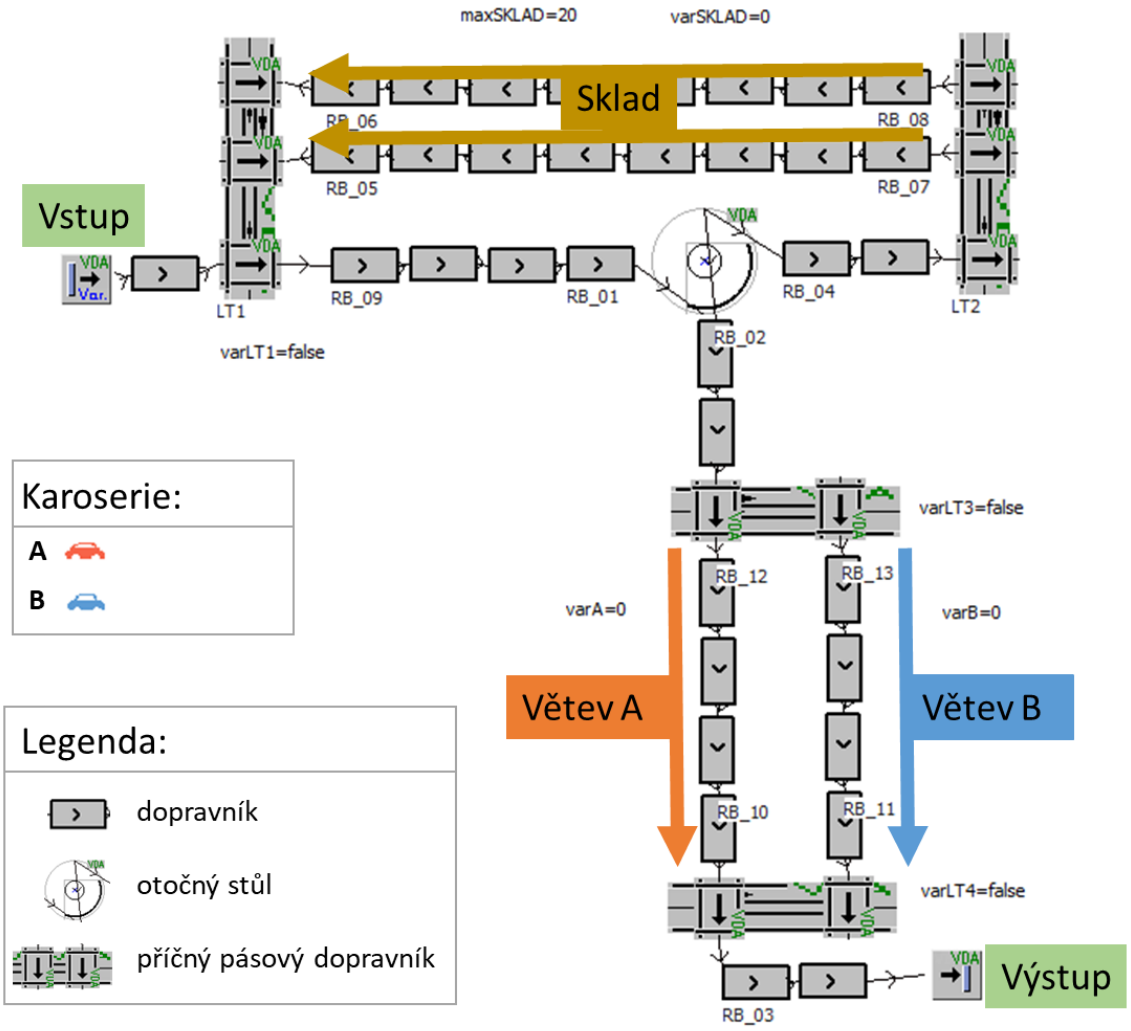
[14] *VISUAL PARADIGM: How to Create a Decision Table?* [online]. [cit. 2022-02-08]. Dostupné z: https://www.visual-paradigm.com/support/documents/vpuserguide/4358/4362/86373_creatingdeci.html

[15] PRESS, Laurence. *Conversion of decision tables to computer programs* [online]. [cit. 2022-02-25].
Dostupné z: https://dl.acm.org/doi/10.1145/364955.364989

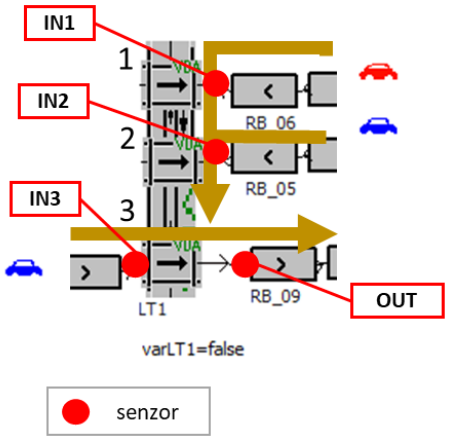
[16] EGLER, J. F. *A procedure for converting logic table conditions into an efficient sequence of test instructions* [online]. [cit. 2022-02-25].
Dostupné z: https://dl.acm.org/doi/10.1145/367593.367595

A Logika řízení simulačního modelu

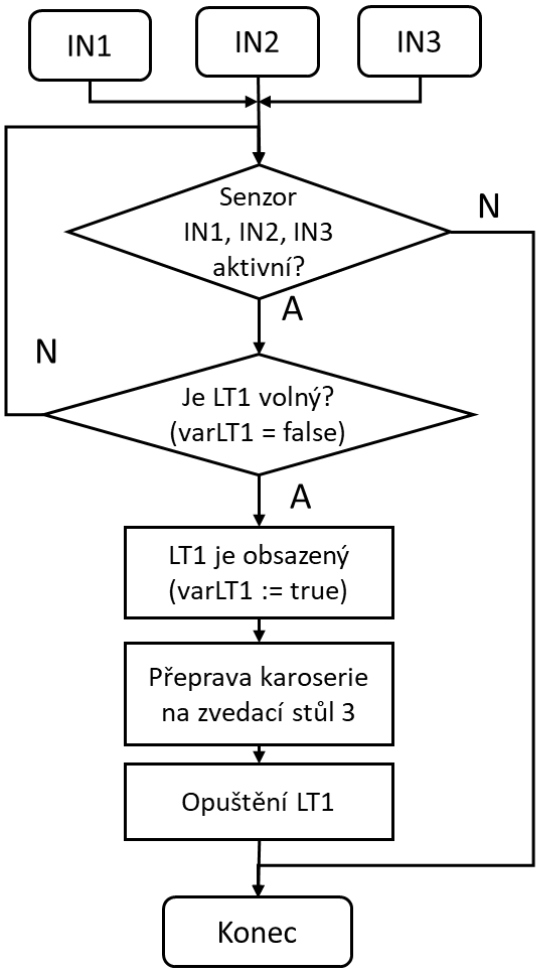
V této příloze je uvedena logika řízení jednotlivých prvků ve formě vývojových diagramů. U každého prvku jsou znázorněny senzory, které řídicí logiku ovlivňují. Vývojové diagramy jsou zobrazeny pod simulačním modelem

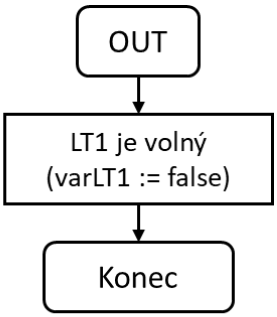


LT1:

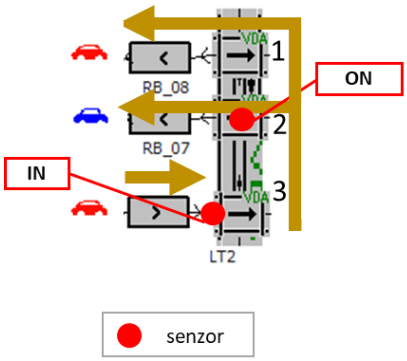


Senzory	
Název	**Prvek**
IN1	Příčný pásový dopravník LT1 (vstup)
IN2	Příčný pásový dopravník LT1 (vstup)
IN3	Příčný pásový dopravník LT1 (vstup)
OUT	Dopravník RB_09 (vstup)

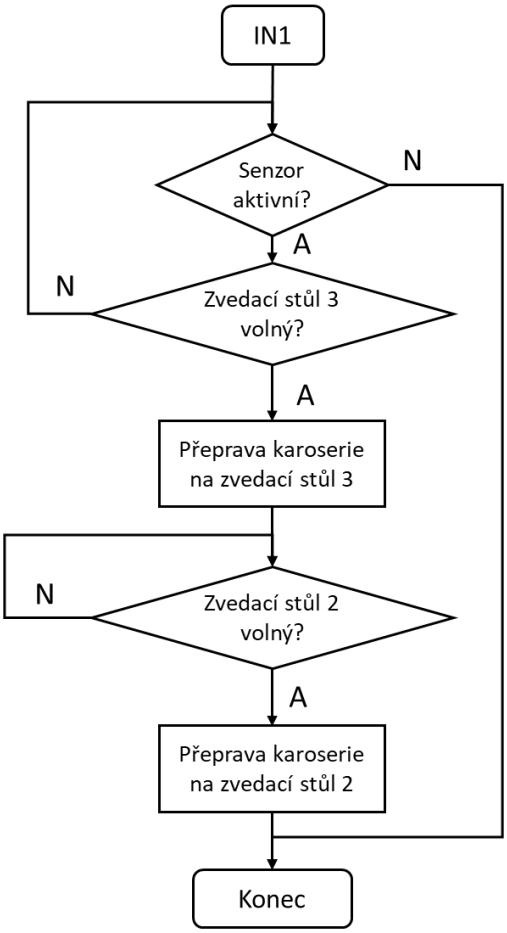


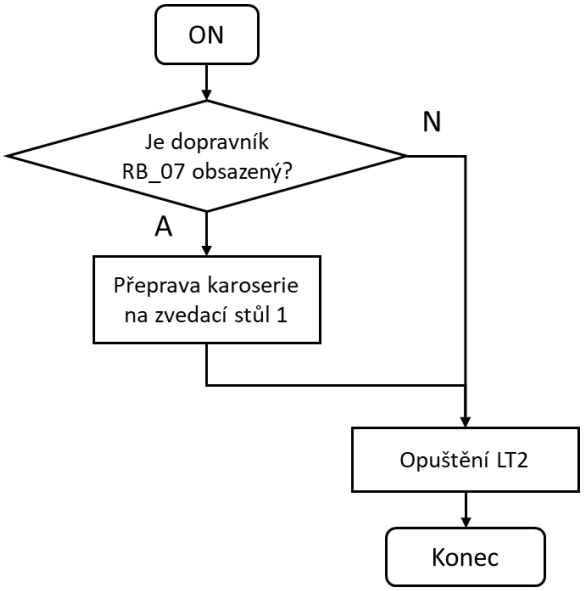


LT2:

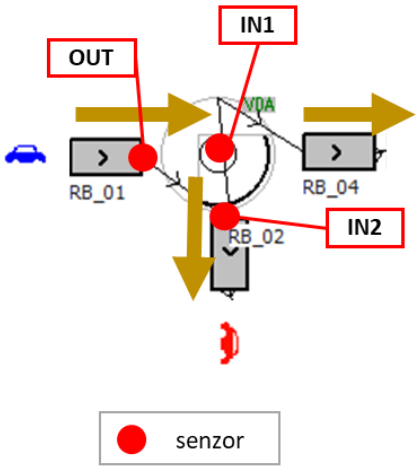


Senzory	
Název	**Prvek**
IN	Příčný pásový dopravník LT2 (vstup)
ON	Příčný pásový dopravník LT2 (zvedací stůl 2)

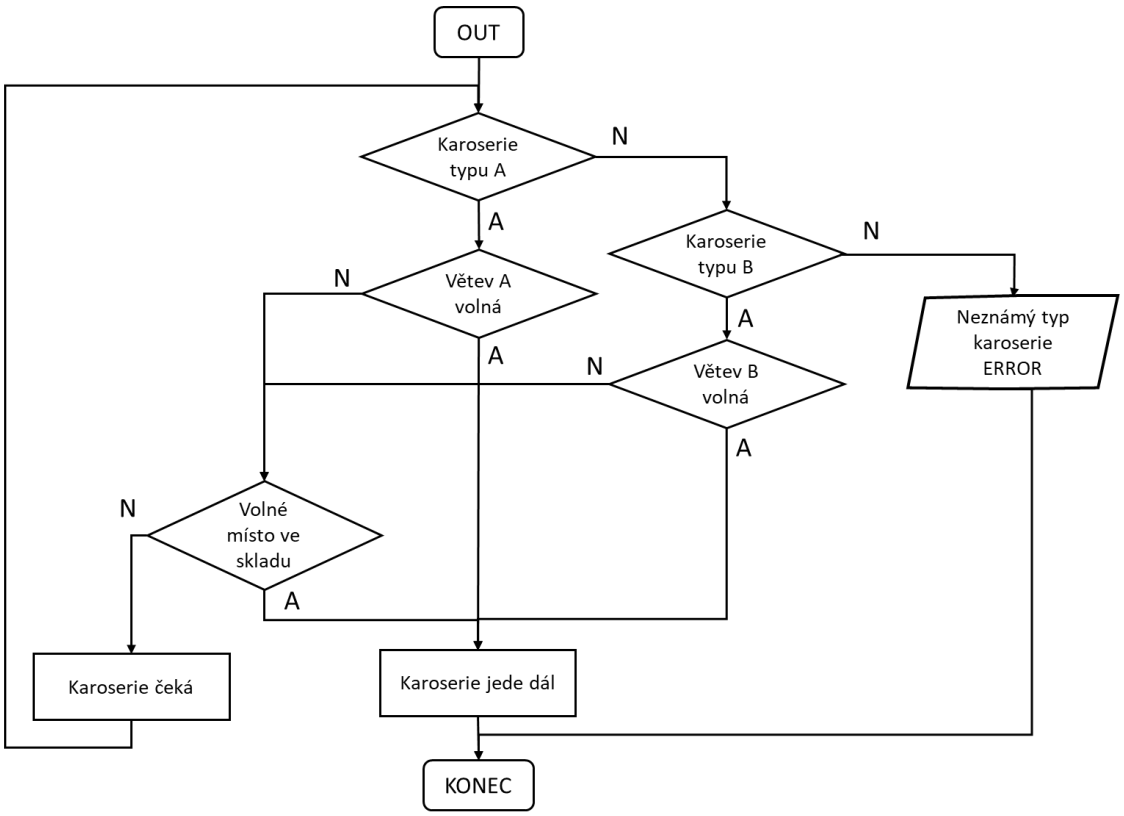


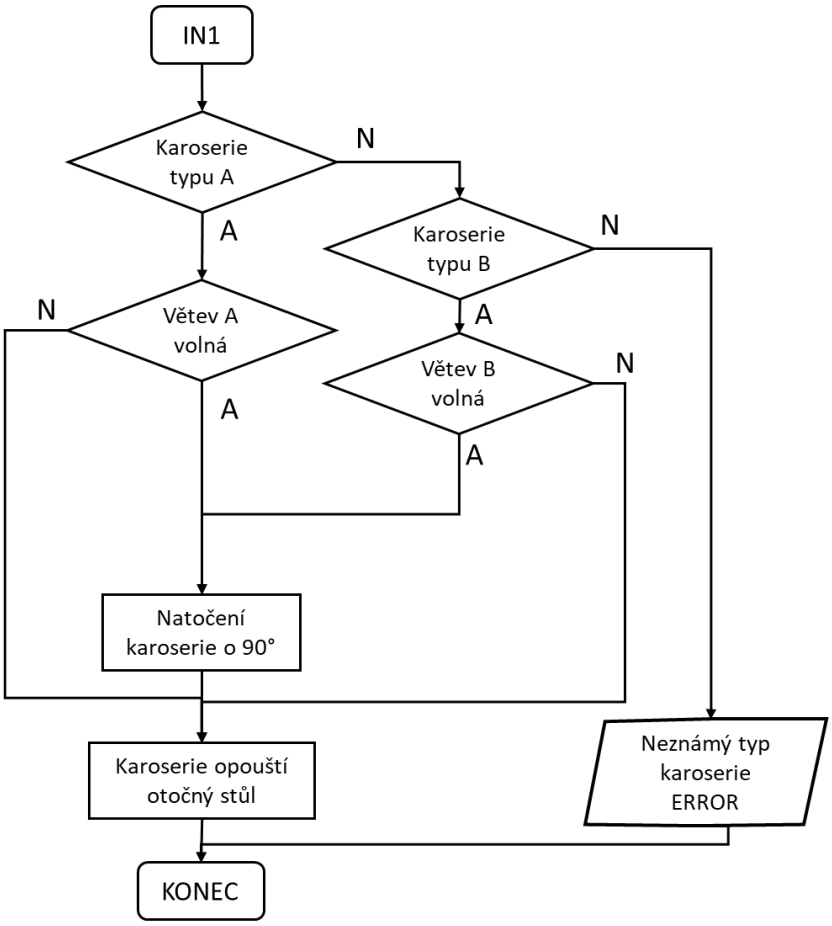


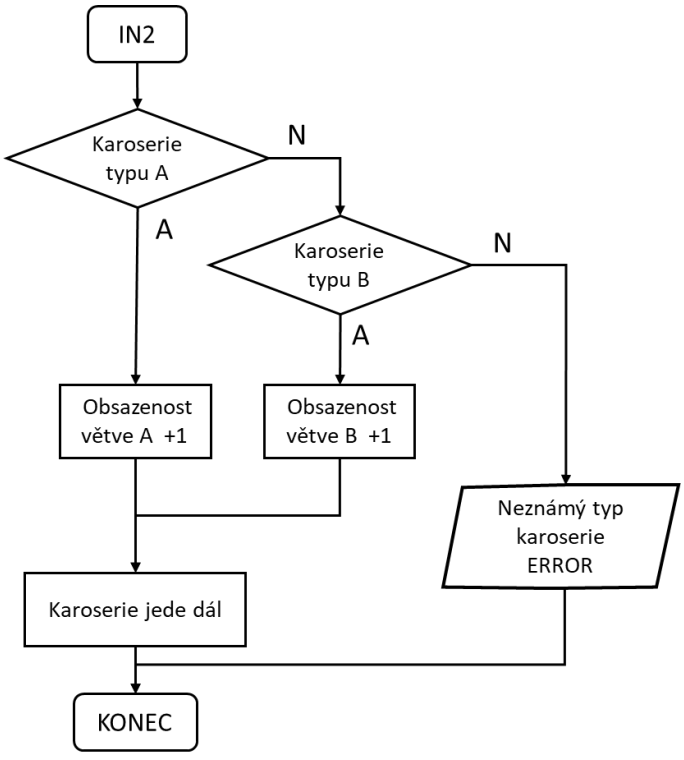
Otočný stůl:



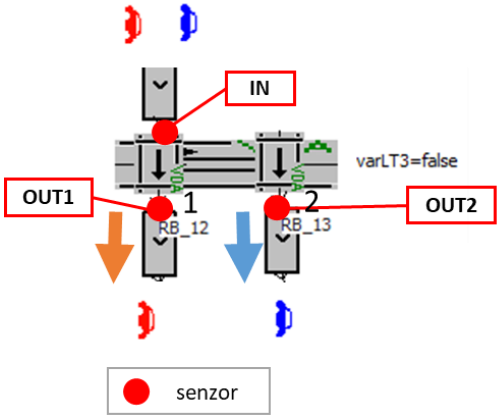
Senzory	
Název	**Prvek**
IN1	Otočný stůl (vstup)
IN2	Dopravník RB_02 (vstup)
OUT	Dopravník RB_01 (výstup)



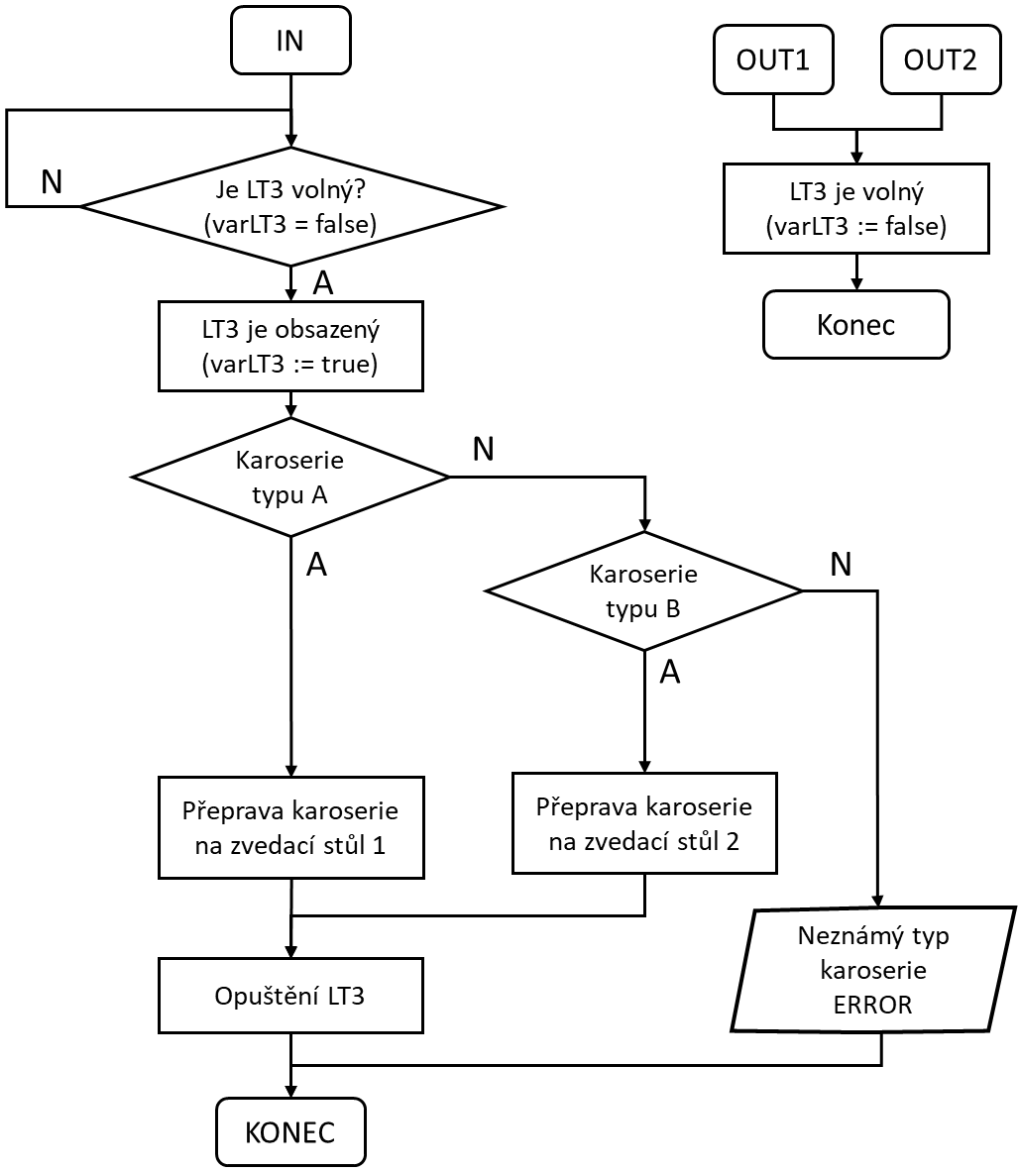

IN1
Karoserie typu A
N
A
Karoserie typu B
N
A
Větev A volná
N
A
Větev B volná
N
A
Natočení karoserie o 90°
Karoserie opouští otočný stůl
Neznámý typ karoserie ERROR
KONEC


IN2
Karoserie typu A
N
A
Karoserie typu B
N
A
Obsazenost větve A +1
Obsazenost větve B +1
Neznámý typ karoserie ERROR
Karoserie jede dál
KONEC

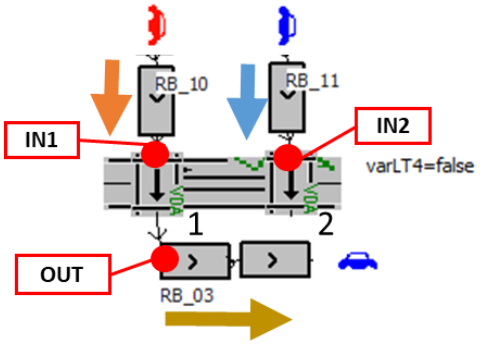
LT3:



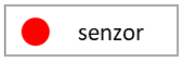
Senzory	
Název	**Prvek**
IN	Příčný pásový dopravník LT3 (vstup)
OUT1	Dopravník RB_12 (vstup)
OUT2	Dopravník RB_13 (vstup)

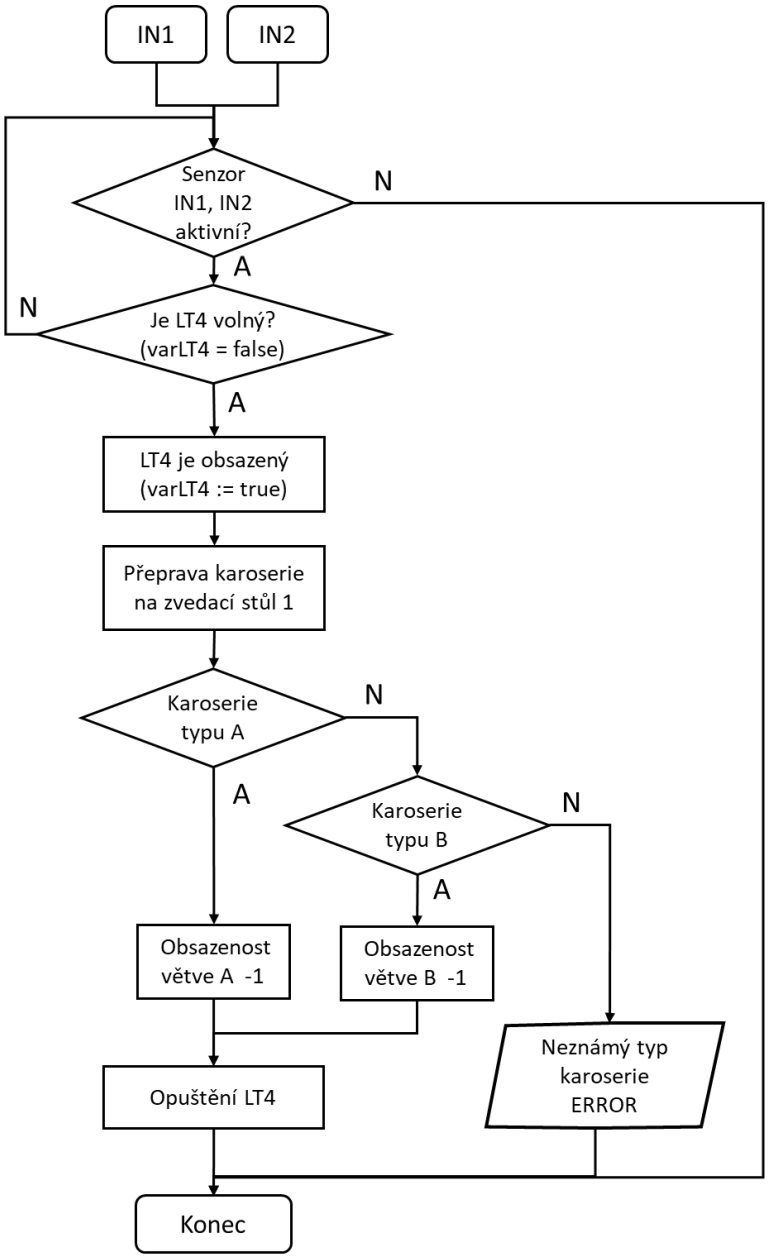


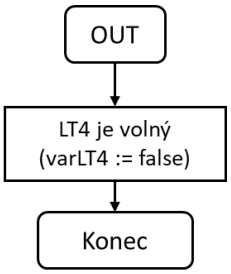
LT4:



Senzory	
Název	**Prvek**
IN1	Příčný pásový dopravník LT4 (vstup)
IN2	Příčný pásový dopravník LT4 (vstup)
OUT	Dopravník RB_03 (vstup)

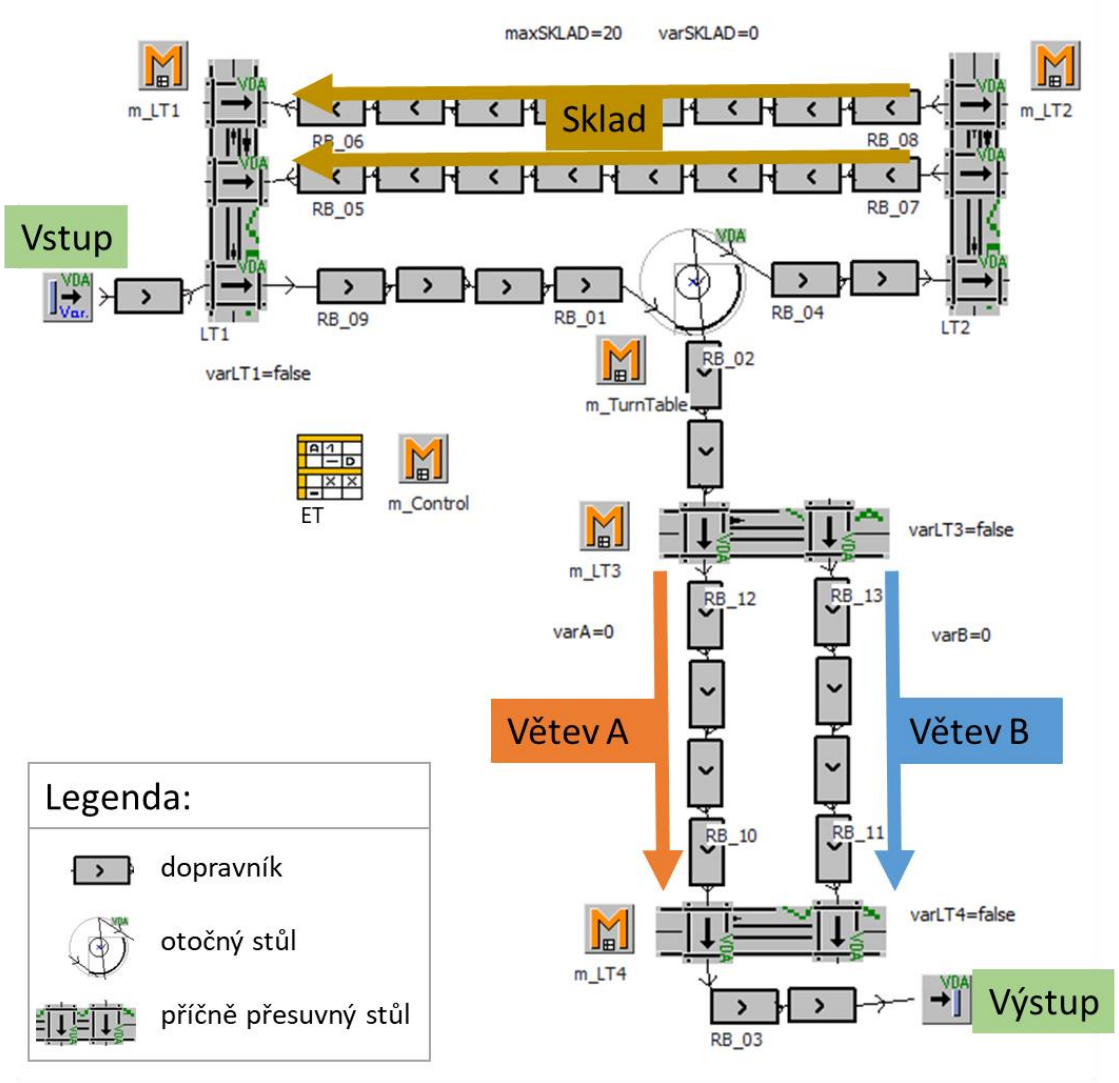






B Simulační model řízený pomocí rozhodovací tabulky

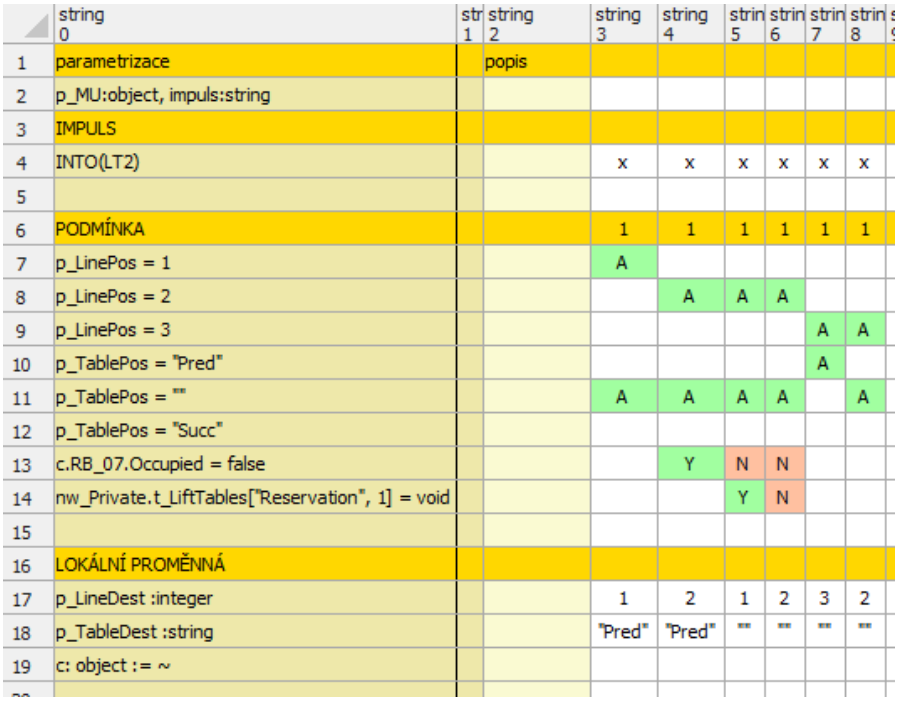
U každého řízeného prvku se nachází jeho řídicí metoda, která je vytvořena rozhodovací tabulkou (ET). Každou z metod lze přetáhnout na ET a zobrazit logiku řízení v tabulkové formě. Pod obrázkem jsou zobrazeny jednotlivé řídicí tabulky daných prvků.



m_LT1:

	string 0	str 1	string 2	strin 3	strin 4	strin 5	string 6	strin 7	strin 8	string 9	strin 10	string 11	string 12	strin 13
1	parametrizace		popis											
2	p_MU:object, impuls:string													
3	IMPULS													
4	INTO(LT1)		posuvny ...	x	x	x	x	x	x	x	x	x	x	
5	RB_09(m_UserInStart)													x
6														
7														
8	PODMÍNKA			1	1	1	1	1	1	1	1	1	1	2
9	p_LinePos = 1			A	A	A	A							
10	p_LinePos = 2							A	A	A				
11	p_LinePos = 3										A	A	A	
12	p_TablePos = "Pred"			A							A	A		
13	p_TablePos = ""				A			A					A	
14	p_TablePos = "Succ"					A	A		A	A				
15	c.varLT1 = false					A	N		A	N	Y	N		
16														
17														
18														
19	LOKÁLNÍ PROMĚNNÁ													
20	p_LineDest :integer				3	1	1	3	2	2	3	3	3	
21	p_TableDest :string				""	""	"Succ"	""	""	"Su...	""	"Pred"	"Su...	
22	c: object := ~													
23														
24	OPATŘENÍ													
25	c.varLT1 := true			X		X			X		X			
26	varLT1 := false													x

m_LT2:



	string 0	str 1	string 2	string 3	string 4	strin 5	strin 6	strin 7	strin 8
1	parametrizace		popis						
2	p_MU:object, impuls:string								
3	IMPULS								
4	INTO(LT2)			x	x	x	x	x	x
5									
6	PODMÍNKA			1	1	1	1	1	1
7	p_LinePos = 1			A					
8	p_LinePos = 2				A	A	A		
9	p_LinePos = 3							A	A
10	p_TablePos = "Pred"							A	
11	p_TablePos = ""			A	A	A	A		A
12	p_TablePos = "Succ"								
13	c.RB_07.Occupied = false				Y	N	N		
14	nw_Private.t_LiftTables["Reservation", 1] = void					Y	N		
15									
16	LOKÁLNÍ PROMĚNNÁ								
17	p_LineDest :integer			1	2	1	2	3	2
18	p_TableDest :string			"Pred"	"Pred"	""	""	""	""
19	c: object := ~								

m_LT3:

	string 0	str 1	string 2	string 3	string 4	strin 5	st 6
1	parametrizace		popis				
2	p_MU:object, impuls:string						
3	IMPULS						
4	INTO(LT3(m_GetDest))			x	x		
5	RB_12(m_UserInStart)					x	
6	RB_13(m_UserInStart)					x	
7							
8	PODMÍNKA			1	1	2	
9	p_LinePos = 1			A	A		
10	p_LinePos = 2						
11	p_TablePos = "Pred"			A	A		
12	p_TablePos = ""						
13	p_TablePos = "Succ"						
14	~.varLT3 = false			A	A		
15	p_MU.Variante1 = "A"			A	N		
16	p_MU.Variante1 = "B"			N	A		
17							
18	LOKÁLNÍ PROMĚNNÁ						
19	p_LineDest :integer			1	2		
20	p_TableDest :string			"Succ"	"Succ"		
21							
22	OPATŘENÍ						
23	~.varLT3 := true			x	x		
24	varLT3 := false					x	

M_LT4:

	string 0	str 1	string 2	string 3	string 4	string 5	string 6	strin 7	st 8
1	parametrizace		popis						
2	p_MU:object, impuls:string -> string								
3	IMPULS								
4	INTO(LT4(m_GetDest))			x	x	x	x		
5	RB_03(m_UserInFinished)							x	
6									
7	PODMÍNKA			1	1	1	1	2	
8	p_LinePos = 1			A	A				
9	p_LinePos = 2					A	A		
10	p_TablePos = "Pred"			A	A	A	A		
11	p_TablePos = ""								
12	p_TablePos = "Succ"								
13	~.varLT4 = false			A	N	A	N		
14									
15	LOKÁLNÍ PROMĚNNÁ								
16	p_LineDest : integer			1	1	1	2		
17	p_TableDest : string			"Su...	"Pred"	"Succ"	"Pred"		
18									
19	OPATŘENÍ								
20	~.varLT4 := true			x		x			
21	varLT4 := false							x	

m_TurnTable:

	string 0	str 1	string 2	strin 3	strin 4	strin 5	strin 6	strin 7
1	parametrizace		popis					
2	p_MU:object,impuls:string							
3	IMPULS							
4	INTO(VDA_MFTable(MF_Turntable))		otocny stul	x	x	x	x	
5								
6	PODMÍNKA			1	1	1	1	
7	p_MU.Variante1 = "A"			A	A	N	N	
8	p_MU.Variante1 = "B"			N	N	A	A	
9	c.varA < 4			A	N			
10	c.varB < 4					A	N	
11								
12	LOKÁLNÍ PROMĚNNÁ							
13	i : integer			3	2	3	2	
14	c: object := ~							
15								
16	OPATŘENÍ							
17	result := i			X	X	X	X	
18								

m_Control:

	string 0	str 1	string 2	strin 3	strin 4	strin 5	strin 6	strin 7	strin 8	strin 9	strin 10	strin 11	strin 12	strin 13	strin 14
1	parametrizace		popis												
2	p_MU:object, impuls:string														
3	IMPULS														
4	RB_02(enter)			x	x										
5	RB_10(finish)					x	x	x	x						
6	RB_11(finish)					x	x	x	x						
7	RB_01(finish)									x	x				
8	RB_04(enter)											x			
9	RB_04(finish)												x		
10	RB_05(finish)													x	
11	RB_06(finish)													x	
12															
13	PODMÍNKA			1	1	2	2	3	3	4	4	5	6	7	
14	p_MU.Variante1 = "A"			Y	N	Y	N	Y	N	A	N				
15	p_MU.Variante1 = "B"			N	Y	N	Y	N	Y	N	A				
16	varA < 4									N					
17	varB < 4										N				
18	RB_01.Stopped = true							Y	Y				A		
19	RB_01.Cont /= void							Y	Y						
20	RB_01.Cont.Variante1 = "A"							Y	N						
21	RB_01.Cont.Variante1 = "B"							N	Y						
22	varLT1 = false														
23	varSKLAD > maxSKLAD									A	A				
24															
25	OPATŘENÍ														
26	varA += 1			X											
27	varB += 1				X										
28	varA -= 1					X									
29	varB -= 1						X								
30	RB_01.Stopped := true									X	X				
31	RB_01.Stopped := false							X	X				X		
32	varLT1 := false														
33	varSKLAD += 1											X			
34	varSKLAD -= 1													X	
35															