Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Multiagentový přístup k řešení logistických problémů víceinstanční průmyslové výroby

Bakalářská práce

Autor: Aleš Lajvr
Studijní obor: Aplikovaná informatika (ai3-k)

Vedoucí práce: RNDr. Petr Tučník, Ph.D.

Hradec Králové listopad 2020

Prohlášení:

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

vlastnoruční podpis

V Hradci Králové dne 21.11.2020 Aleš Lajvr

Poděkování:

Děkuji vedoucímu bakalářské práce RNDr. Petru Tučníkovi, Ph.D. za metodické vedení práce, ochotu a trpělivost.

Anotace

V současné době probíhá ve všech oborech lidské činnosti digitalizace. V průmyslovém odvětví je tento proces označován jako čtvrtá průmyslová revoluce. Tzv. chytré továrny mají přinést vyšší produktivitu výroby při současném snížení nákladů a časové náročnosti.

Vědeckými studiemi bylo dokázáno, že v dodavatelských řetězcích i velkých továrnách dochází k neefektivnímu hospodaření, a to především z důvodu nedostatečného sdílení informací. Podle vědců je optimalizace dodavatelského řetězce jako celku důležitější, než optimalizace jeho částí. Řešením komunikačních problémů a nástrojem pro úspěšnou optimalizaci se mohou stát multiagentové systémy (MAS). Mezi hlavní přednosti MAS patří kooperace agentů a racionální rozhodování.

V rámci této práce byl vytvořen model dodavatelského řetězce umožňující simulaci činnosti výrobního řetězce včetně logistiky. Díky simulaci se podařilo ověřit vliv logistických chyb na produktivitu jednotlivých výrobních linek i celého řetězce. V rámci experimentů se ukázalo, že dovezením surovin od záložního dodavatele, navýšením rezervy surovin ve skladu a zvýšením produktivity se podařilo vliv výpadku zásobování úplně eliminovat.

Annotation

Title: Multi-Agent Approach to Logistic Problems in Multi-Plant Production Logistics

Digitalization is currently taking place in all fields of human activity. In the industrial sector, this process is referenced as the fourth industrial revolution. The so-called smart factories are supposed to bring higher production productivity while reducing costs and time.

Scientific studies have shown that supply chains and large factories are inefficient, mainly due to insufficient information sharing. According to scientists, optimizing the supply chain altogether is more important than optimizing its parts. Multi-agent systems (MAS) can be the solution to communication problems and a tool for successful optimization. The main advantages of the MAS include agent cooperation and rational decision-making.

Within this work, a supply chain model has been created to simulate the activities of the production chain, including logistics. Due to the simulation, it has been possible to verify the effect of logistics errors on the productivity of individual production lines and the entire chain. The experiments showed that by importing raw materials from the backup supplier, increasing the reserve of raw materials in the warehouse, and increasing productivity, the effect of the supply outage was eliminated completely.

Obsah

Seznam zkratek

Zkratka	Popis
ACL	Agent Communication Language
AGV	Automated Guided Vehicle
CARP	Context Aware Route Planning
CDPS	Cooperative Distributed Problem Solving
CFP	Call For Proposals
CPLEX	C-Simplex (Simplex algorithm that is written in C)
CPPS	Cyber Physical Production System
CPS	Cyber Physical Systems
CTV	Cellular Transport Vehicle
DARPA	Defense Advanced Research Projects Agency
FiFo	First In First Out
FIPA	The Foundation for Intelligent Physical Agents
GIS	Geographic Information System
GUI	Graphical User Interface
IBM	International Business Machines Corporation
IEEE	The Institute of Electrical and Electronics Engineers
KQML	Knowledge Query and Manipulation Language
KSE	Knowledge Sharing Effort
MAS	Multi-Agent System
MES	Manufacturing Execution Systems
OEE	Overall Equipment Effectiveness
PEAS	Performance, Environment, Actuators, Sensors
PL	Production Logistics
PLE	Personal Learning Edition
PO-Agent	Production Order agent
POM-Agent	Production Order Management agent
SPL	Synchronized Production and Logistics
SQL	Structured Query Language
Ubi-SPL	Synchronized Production and Logistics via Ubiquitous Technology
UML	Unified Modelling Language

Seznam obrázků

Seznam tabulek

1 Úvod

Klesající cena a stoupající výkon výpočetních systémů stále více ovlivňují všechny obory lidské činnosti. Současně dochází ke stále větší míře miniaturizace počítačů i elektroniky obecně. Vědci vynalézají přesnější a rychlejší senzory i aktuátory. Všechny tyto skutečnosti umožňují vývoj autonomních robotů a systémů, jejichž tvůrci si kladou za cíl nejen usnadnění těžké manuální práce ve výrobních provozech, ale i pomoc při rozhodování vrcholového managementu. Díky zpřesnění a zrychlení rozhodování se zvyšuje produktivita, snižuje míra využití lidské práce a tím klesá úrazovost a snižuje se cena produktů při zachování, nebo i zvýšení kvality.

V průmyslovém odvětví vznikla iniciativa Průmysl 4.0, což je celý soubor technologií, protokolů a jejich vzájemného propojení s cílem zefektivnit průmyslovou výrobu. Jednou z částí, kterým se průmysl 4.0 věnuje, je využití autonomních systémů ve výrobním procesu a procesech řízení. Autonomní systémy mohou výrazně snížit riziko plynoucí z chybného rozhodnutí, protože na rozdíl od lidské obsluhy provádějí svá rozhodnutí výhradně racionálně a nemůže je ovlivnit stres, přepracování, nebo emoce.

Tato práce se zabývá využitím multiagentových systémů (MAS) v logistice víceinstanční průmyslové výroby a klade si za cíl posouzení míry vlivu výpadku logistiky na jednotlivé výrobní instance a dodavatelský řetězec jako celek. V rámci práce bude vytvořen model MAS, který umožní simulaci činnosti výrobního řetězce, včetně externích dodavatelů, zákazníků a logistiky. Simulaci bude možné spustit v režimu bez poruchy, případně s poruchou vybrané výrobní instance. Na základě výsledků simulace uspořádaných do grafů by mělo být možné vyhodnotit vliv různých druhů chyb a úspěšnost opatření, která mají zabránit šíření chyby se všemi důsledky.

2 Teoretická část a základní pojmy

2.1 Agenty

Definice agentu není jedna univerzálně akceptovatelná a záleží na situačním kontextu, proto se u různých autorů může lišit. Jak uvádí (Russell a Norvig 2016) agent je něco, co vnímá a jedná v prostředí. Autoři dále uvádějí, že funkce agentu specifikuje akci podniknutou agentem v odezvě na jakoukoli sekvenci vjemů. (Wooldridge 2009) agent definuje jako počítačový systém situovaný v určitém prostředí, který je schopen samostatného jednání v tomto prostředí za účelem splnění svých delegovaných cílů. (Kubík 2004) uvádí: *„Agent je entita zkonstruovaná za účelem kontinuálně a do jisté míry autonomně plnit své cíle v adekvátním prostředí na základě vnímání prostřednictvím senzorů a prováděním akcí prostřednictvím aktuátorů“*. Autor dále uvádí: *„Agent přitom ovlivňuje podmínky v prostředí tak, aby se přibližoval k plnění cílů“*. (Tanajura et al. 2015) definuje agent jako výpočetní systém umístěný v prostředí, který je schopný v tomto prostředí samostatně jednat za účelem dosažení navržených cílů. (Lu et al. 2019) definuje agenty jako nezávislé jednotlivce, kteří mohou průběžně provádět autonomní rozhodnutí a akce v distribuovaných, nebo kolaborativních systémech.

Uvedené popisy agentů mají jeden společný rys, jsou abstraktní. Důvodem je jednak snaha autorů odstínit veškeré komponenty, bez kterých by sice agent nemohl fungovat (hardware, operační systém, programovací jazyk atd.), které však mohou být bez problému nahrazeny jinými komponentami. Není důležité ani to, zda se jedná o stejnou HW konstrukci, či platformu OS. Jinými slovy, agent by měl být nezávislý na použitém hardware i operačním systému. Benefitem, který vyplývá z těchto definic je to, že agent může být mobilní v tom smyslu, že je schopný migrovat mezi různými platformami. Tím lze optimalizovat dosažitelnost externích zdrojů (konektivita se senzory, kooperujícími agenty atd.), které agent potřebuje pro právě vykonávanou, nebo plánovanou akci. Díky výše zmíněným schopnostem agentů lze rovněž distribuovat úkoly dalším agentům, které mají lepší předpoklady pro úspěšné splnění těchto úkolů, nebo využít decentralizované řízení agentů.

Kromě toho lze optimalizovat využití zdrojů hostitele (CPU, paměť atd.). Pokud agent zjistí, že má nedostatek zdrojů, tak si může vyhledat jiného hostitele, který je aktuálně méně vytížený, nebo má více zdrojů. Agenty rozdělujeme podle jejich schopností na následující typy:

- Reaktivní agenty: Jedná se o základní a nejjednodušší typ agentu. Reaktivní agenty podle (Kubík 2004) nemají žádnou umělou inteligenci, umí pouze reagovat na aktuální stav prostředí, na svůj vnitřní stav, případně na kombinaci obojího. Mezi výhody reaktivních agentů patří podle (Wooldridge 2009) jednoduchost, hospodárnost, výpočetní sledovatelnost a robustnost vůči selhání. Tento typ agentů je podle autora vhodný pro řešení jednoduchých úkolů, kde není vyžadováno zapamatování modelu prostředí. Autor dále uvádí, že agenty musí mít pro určení přijatelné akce dostatek informací ve svém vnitřním stavu. To podle autora způsobuje, že v případě rozhodování mají agenty k dispozici pouze krátkodobý pohled.
- Deliberativní agenty: Mají stejné schopnosti jako reaktivní agenty, ale navíc jsou rozšířeny o schopnost autonomního rozhodování. Mohou se učit a plánovat cestu ke splnění cílů na základě již naučených zkušeností. Jejich schopnosti a dovednosti se tedy mohou rozvíjet. Jak uvádí (Kubík 2004) *„Deliberativní (uvažující) agent si uchovává symbolické reprezentace prostředí a vnitřních stavů, na jejichž základě vytváří plány pro dosahování svých cílů“*.
- Sociální agenty: Slouží k vybudování sociálních sítí mezi agenty. Pro případné využití kooperace si uchovávají informace o ostatních agentech v síti, tj. jejich označení (jméno), adresu, aktuální stav a schopnosti. (Kubík 2004) tuto definici doplňuje o informaci, že komunikovat mohou i reaktivní agenty, v případě sociálního agentu se však jedná o vyšší komunikační jazyk.
- Hybridní agenty: Kombinují vlastnosti některých, případně všech výše zmíněných druhů agentů

2.2 Multiagentové systémy

Multiagentové systémy (zkráceně MAS), jsou tvořeny více agenty operujícími ve společném prostředí. Agenty jsou navržené tak, aby fungovaly nezávisle, zároveň však v případě potřeby na sebe mohou vzájemně působit, toto jsou klíčové vlastnosti pro použití v MAS. Podle (Russell a Norvig 2016; Kubík 2004) existuje několik typů vztahů mezi agenty:

- Konkurence spočívá v maximalizaci výkonosti jednoho agentu, nebo skupiny agentů na úkor výkonosti jiného agentu, nebo skupiny agentů. (Russell a Norvig 2016) uvádějí jako příklad konkurence šachový zápas dvou agentů. Podle autorů se každý z agentů snaží maximalizovat svou výkonnost, což podle šachových pravidel zároveň minimalizuje výkonnost protivníka.
- Koordinace je podle (Kubík 2004) *„proces probíhající v MAS, kterým se dosahuje takové propojení jednotlivých komponent v systému, které umožňuje řešení problému dosažením decentralizace vykonávaných úkolů a někdy i řídícího procesu“*. Autor dále uvádí, že *„koordinace je ve společenství agentů důležitá pro dosahování cílů na úrovni celého systému“*. Koordinace může podle autora: *„být centralizovaná, ve které agent s řídícím postavením určuje role a úlohy ostatních agentů, nebo decentralizovaná“*.
- Kooperace je podle (Kubík 2004) *„vyšší formou koordinace“*, autor dále uvádí, že se jedná o *„řízenou formu koordinace s účelovým uspořádáním agentů ve skupině s cílem dosáhnout společného řešení problému, nebo konfliktu“*. (Russell a Norvig 2016) uvádějí jako příklad kooperace agenty v roli řidičů taxi. Podle autorů jsou vztahy mezi agenty částečně kooperativní, což přispívá k minimalizaci možných kolizí a tím dochází k maximalizaci výkonnosti všech agentů. Zároveň se však podle autorů jedná o částečně konkurenční vztahy, například proto, že dva agenty nemohou obsadit jedno parkovací místo.

Multiagentové plánování je podle (Russell a Norvig 2016) nezbytné, pokud v prostředí existují další agenty, mezi kterými existují kooperativní, nebo konkurenční vztahy. Podle autorů mohou být vytvořeny společné plány, ale musí být

rozšířeny o nějakou formou koordinace, aby se dva agenty mohly dohodnout na tom, který společný plán provést.

Autoři (Tanajura et al. 2015; Lu et al. 2019) ve svých pracích uvádějí, že každá část, nebo charakteristika systému může být reprezentována agentem, který jedná nezávisle a zároveň kooperativně. Kromě výše uvedené definice, (Lu et al. 2019) MAS definuje jako volně distribuovaný systém, který sestává z mnoha agentů. Autor dále uvádí, že MAS může činit přiměřená opatření pro cíle a zdroje, takže v případě sporu o zdroje, nebo při konfliktu, může každý agent koordinovat své znalosti, cíle, strategie atd., za účelem eliminování konfliktů. Pokud má jeden agent nedostatek zdrojů pro vykonání nějaké akce a není schopný sám zajistit nápravu, tak může požádat o spolupráci jiný agent. Cíl, pro jehož dokončení nemá agent dostatek zdrojů, může předat k řešení buď celý, nebo jej může rozdělit na více částí, a zapojit do spolupráce jeden, nebo i více dalších agentů.

Další výhodou MAS je škálovatelnost, pokud stávající systém nestíhá vyřizovat požadavky, můžeme za provozu přidávat do systému další agenty. Pokud naopak některé agenty již nejsou potřebné pro funkčnost systému, lze je odebrat rovněž za provozu. Významnou výhodou agentově orientovaného přístupu je podle (Tucnik et al. 2017) možnost vybudování komplexního systému založeného na spolupráci jednotlivých agentů, díky přístupu zdola nahoru. To podle autorů umožňují agenty s jednoduchou modulární konstrukcí, které mohou opakovaně využít základní modely. Autoři dále uvádějí, že složitost takového systému může rychle růst díky tomu, že se do modelu postupně začleňují další podrobnosti. Podle autorů je nutné udržet složitost systému v kontrolovatelných mezích použitím vysoce modulárního přístupu, tedy návrhem agentů z podobných „stavebních bloků“.

Standardizací v oblasti agentů a multiagentových systémů včetně jejich interoperability vzhledem k ostatním technologiím se zabývala organizace FIPA (The Foundation for Intelligent Physical Agents). Podle (Wooldridge 2009) započala FIPA svou činnost na vývoji standardů pro agentní systémy v roce 1995. V roce 2005 byla podle (FIPA 2019) organizace přijata jako výbor pro standardy IEEE Computer Society. Podle (Wooldridge 2009) byl hlavní náplní práce organizace FIPA vývoj komunikačního protokolu ACL (Agent Communication Language), viz kapitola 2.2.1.

V současné době podle (FIPA 2019) organizace není aktivní, její sbírka norem je však přístupná všem zájemcům.

2.2.1 Komunikace a koordinační mechanismy v MAS

Pro zajištění vzájemné synchronizace agentů je nutné, aby spolu dokázaly komunikovat. Podle (Wooldridge 2009) je synchronizace agentů nezbytná, zejména pokud existuje možnost vzájemné kolize agentů. Agenty podle autora nemohou vynutit vykonání nějaké akce jinými agenty, ani zapsat data do vnitřního stavu jiných agentů. Podle autora však mohou vykonat komunikační akce, ve snaze vhodně ovlivnit jiné agenty. Předání požadavku na vykonání nějaké akce však nemusí nutně skončit úspěchem, protože jsou agenty autonomní. Záleží především na tom, jestli je akce, kterou požaduje agent v roli žadatele ve shodě s cíli agentu, kterému je požadavek adresován.

Komunikace mezi agenty může být přímá, nebo nepřímá. Nejzákladnější forma koordinace je zajištěna prostřednictvím (nepřímé) reaktivní komunikace, která probíhá zanecháním značek (zpráv) v prostředí. Tento způsob nepřímé komunikace se podle (Kubík 2004) *„nazývá stigmergie podle způsobu komunikace společenského hmyzu“* a obvykle se týká reaktivních agentů. Reaktivní komunikace má podle autora následující výhody: *„Systém je robustní, protože chybovost akce robota nezpůsobí fatální selhání celého systému“*, *„Systém se vyznačuje posloupnou degradací funkcionality, protože výpadek jednoho, nebo několika robotů nezpůsobí krach celé mise“*, *„Systém je adaptivní a flexibilní. Změny v prostředí mají jen malý vliv na jeho funkcionalitu“*, *„Agenty se vyznačují minimální množinou symbolické reprezentace“* a *„Tvorba takového systému se ukazuje ekonomicky výhodná kvůli jednoduché hardwarové i softwarové implementaci agentů“*.

V softwarovém inženýrství se používají spíše vyšší formy (přímé) komunikace, respektive vyšší komunikační protokoly. Jako příklad lze uvést jazyk KQML (Knowledge Query and Manipulation Language), který podle (Wooldridge 2009) vznikl z jazyka KSE (Knowledge Sharing Effort), jehož vytvoření financovala organizace DARPA na začátku 90. let 19. století. Jazyk KQML je podle autora založený na zprávách, každá zpráva se skládá z jednoho příkazu a několika

parametrů. Autor dále uvádí, že parametry slouží především pro definici obsahu zprávy, identifikaci příjemce a odesilatele a identifikaci odpovědi v případě konverzace. Jazyk KQML byl sice komunitou MAS přijat a podle (Wooldridge 2009) vzniklo i několik implementací založených na tomto jazyce, navzdory těmto úspěchům byl však následně jazyk z mnoha důvodů kritizován. Mezi tyto důvody podle autora patří: nebyla pevně stanovená základní sada příkazů, nebyl přesně definovaný mechanismus přenosu zpráv, nebyla přesně stanovená sémantika, v jazyce chyběla třída závazkových příkazů a sada příkazů byla příliš velká. Autor dále uvádí, že výše zmíněné nevýhody pak v konečném důsledku zapříčinily nemožnost spolupráce mezi agenty s různými implementacemi KQML, což vedlo k vývoji nového jazyka ACL.

Jazyk ACL (Agent Communication Language) byl vytvořen organizací FIPA v roce 1999. ACL je podle (Wooldridge 2009) velmi podobný jazyku KQML, definuje vnější jazyk pro zprávy, dále definuje 20 příkazů pro určení zamýšlené interpretace zpráv a nezakládá žádný specifický jazyk pro obsah zprávy. Autor dále uvádí, že výše zmíněná podobnost obou jazyků spočívá ve stejné struktuře zpráv a oba jazyky mají rovněž velmi podobné parametry. Nejdůležitější rozdíl mezi oběma jazyky podle autora spočívá v odlišné sadě příkazů.

V práci (Lu et al. 2019) jsou způsoby komunikace rozdělené do dvou kategorií. V prvním případě se podle autorů jedná o odeslání informace, kdy agent odesilatel pošle informaci agentu příjemci. Ve druhém případě se podle autorů jedná o vyžádání služby, kdy si agent odesilatel vyžádá od agentu příjemce zaslání informace, kterou potřebuje ke své činnosti. Autoři dále uvádějí, že je použita přímá komunikace a každý agent disponuje funkcí odesílání a příjmu zpráv. Podle autorů musí být formát zpráv jednotný a každá zpráva musí být zapsána podle pravidel, aby mohla být efektivně zpracována.

Spolupráci agentů v MAS podle (Wooldridge 2009) popisuje proces nazvaný CDPS (Cooperative Distributed Problem Solving). Podle autora je nutné v CDPS řešit následující činnosti:

1) rozdělení problému neboli dekompozici na podproblémy, které lze distribuovat mezi jednotlivé agenty
2) vyřešení podproblémů jednotlivými agenty, přičemž agenty mohou mezi sebou sdílet informace
3) zajištění efektivní syntézy řešení problému z výsledků podproblémů
4) optimalizace aktivit agentů řešících společně zadaný problém za účelem maximalizace měřítek soudržnosti
5) zajištění koordinace činnosti agentů, čímž se zabrání destruktivním interakcím a zároveň se maximalizuje účinnost

Autor zavádí pojmy sdílení úkolů a sdílení výsledků. Sdílení úkolů spočívá, podle autora, v dekomponování problému a přidělení podproblémů jednotlivým agentům. Sdílení výsledků spočívá podle autora v tom, že agenty poskytují ostatním agentům relevantní informace a to proaktivně, nebo na vyžádání.

Koordinace je podle (Wooldridge 2009) nezbytná, pokud činnosti, které agenty mohou provádět, mohou jakýmkoliv způsobem interagovat. Jako příklad autor uvádí dvě osoby, které chtějí projít dveřmi a opustit místnost. Do dveří se však vejde pouze jedna z osob, a proto tyto osoby musí svou činnost koordinovat. V MAS je situace analogická, osoby lze označit jako agenty a dveře jako společný zdroj.

Při řešení problémů v MAS je podle (Wooldridge 2009) kromě koordinace rovněž nezbytné plánování a synchronizace. Autor uvádí následující možnosti:

- Centralizované plánování pro distribuované plány: „Vedoucí“ agent vytvoří plán pro skupinu „podřízených“ agentů, ve kterém je definováno rozdělení a uspořádání práce. Poté „vedoucí“ agent distribuuje plán jednotlivým „podřízeným“ agentům, které mají vykonat svou část plánu.
- Distribuované plánování: Skupina agentů „plánovačů“ spolupracuje na vytvoření společného centralizovaného plánu. Jednotlivé agenty se obvykle specializují v různých aspektech celkového plánu, jejich úkolem je však pouze vytvoření plánu, provádět jej nebudou.
- Distribuované plánování distribuovaných plánů: Skupina agentů spolupracuje při tvorbě individuálních akčních plánů a dynamicky koordinují

své činnosti. Agenty mohou mít své vlastní zájmy, potenciální problémy s koordinací je proto nutné řešit vyjednáváním.

2.2.2 Přidělování zdrojů agentům

V případě existence omezených zdrojů, které může a má zájem využít více než jeden agent, je vhodné dosáhnout dohody o způsobu přidělení těchto zdrojů agentům. Pro účely výběru vhodných kandidátů pro přidělení zdrojů je podle (Wooldridge 2009) efektivní použití aukcí. Autor dále uvádí, že při použití aukcí lze určit agenty, které dané zdroje nejvíce oceňují. Kromě toho je podle autora možné u některých typů aukcí zjistit i skryté pravdy o uchazečích. Autor dále uvádí, že aukce probíhají mezi agentem označovaným jako dražitel a skupinou agentů označovaných jako uchazeči. Podle (Kubík 2004) je *„základní myšlenkou aukce alokace zdrojů (zboží, služeb, řešení, problému apod.) na základě jejich ohodnocení nabídkou a poptávkou“*. Podle (Wooldridge 2009) je pro většinu typů aukcí typické, že se dražitel snaží maximalizovat prodejní cenu, zatímco uchazeči se snaží prodejní cenu minimalizovat.

Podle (Wooldridge 2009) existuje více faktorů ovlivňujících aukční protokol i strategii, přičemž nejdůležitějším z nich je hodnota zdroje určeného k dražbě. Podle autora může mít stejný zdroj pro různé uchazeče různou vnitřní hodnotu. Vnitřní hodnota určuje cenu, kterou je konkrétní uchazeč schopný a ochotný investovat. Kromě vnitřní hodnoty existuje podle (Wooldridge 2009) i hodnota veřejná a hodnota korelovaná (kombinace hodnoty vnitřní a veřejné).

Autor dále rozděluje aukční protokoly na dva typy. První typ je nazvaný „First-price auction“ a vítězí v něm uchazeč s nejvyšší nabídkou, přičemž získává zdroj za cenu, kterou nabídl. Druhý typ je nazvaný „Second-price auction“ a vítězí v něm opět uchazeč s nejvyšší nabídkou, v tomto případě však získává zdroj za cenu druhé nejvyšší nabídky.

Další možnost nastavení aukčního protokolu spočívá podle autora ve způsobu zveřejnění nabídek. Pokud jsou všechny nabídky známé všem uchazečům, jedná se o otevřenou aukci nazývanou „Open cry“. V opačném případě, kdy uchazeči

nejsou schopní určit nabídky ostatních se jedná o uzavřenou aukci nazývanou „Sealed bid“.

Aukce lze podle autora dále dělit podle mechanismu, kterým nabízení probíhá. Aukce s jediným kolem nabídek se podle autora nazývají „One-shot auction“. Aukce, která začíná nízkou cenou a ta je postupně zvyšována dalšími nabídkami se nazývá „Ascending auction“. Aukce, která začíná vysokou cenou a ta je postupně snižována se nazývá „Descending auction“.

Podle (Wooldridge 2009; Kubík 2004) existují následující typy aukcí odlišující se vzájemně pravidly protokolu a strategiemi uchazečů:

- Anglická aukce: „Pravidla aukce stanovují pro účastníky možnost nasadit jakoukoliv cenu (pokud není stanovena dolní hranice) s tím, že všechny agenty znají nabídku ostatních. Aukci moderuje aukcionář a vyhrává ji agent s nejvyšší nabídkou, pokud žádný agent již cenu nezvyšuje. Strategie agentů je založena na vnitřní hodnotě zdroje a také na možnostech ostatních agentů. Vítězná strategie spočívá ve zvyšování nabídky o malé intervaly až do momentu dosažení vnitřní hodnoty. Agent je přitom ovlivňován historií nabídek v dané aukci.“
- Holandská aukce: „V této jednostranné aukci aukcionář vyvolává cenu zdroje na uměle vysoké výši a postupně ji snižuje až do momentu, kdy jeden z agentů nabídne cenu stanovenou aukcionářem. Tento agent aukci vyhrává s danou cenou za zdroj. Strategie agentů je založena na vnitřní hodnotě zdroje a neexistuje dominantní strategie.“
- First-price sealed bid: „V tomto typu jednostranné aukce agenty neznají nabídky ostatních agentů a mají možnost udělat nabídku jenom jednou. Vyhrává agent s nejvyšší cenou nabídky. Strategie účastníků odráží v podstatě jenom vnitřní hodnotu aukčního zdroje s případným odhadem nabídky ostatních agentů. Pro nedostatek informací o ostatních agentech je těžké identifikovat vítěznou strategii.“
- Vickreyova aukce: „Podobně jako v předchozí aukci i tento typ je založen na jednom kole nabídek, přičemž agenty neznají nabídku ostatních agentů. Aukci vyhrává agent s nejvyšší nabídkou, ale zdroj je v této aukci přidělen za

cenu druhé nejvyšší nabídky. Agent, který chce vyhrát, nemůže cenu zdroje podceňovat (protože by byl poražen agentem nabízejícím víc), ale ani přeceňovat (protože by hrozilo, že druhá nejvyšší nabídka překročí jeho vlastní ocenění daného zdroje).“

- Oboustranná aukce: „Tento typ aukce je založen na aktivní účasti strany nabídky i poptávky. Obě strany uvádějí cenu zdroje, za kterou chtějí daný zdroj prodat – pokud možno s co nejvyšší cenou (resp. nakoupit – pokud možno za co nejnižší cenu). Cílem aukcionářů je najít co nejvyšší objem obchodů, při kterých cena poptávky převyšuje cenu nabídky. Jedná se o vysoce dynamickou aukci se složitými strategiemi nakupujících i prodávajících agentů.“

2.2.3 Úkolová prostředí

Úkolová prostředí jsou podle (Russell a Norvig 2016) dostatečně do detailu popsané problémy, pro které racionální agenty představují řešení. Popis úkolových prostředí podle autorů sestává z měření výkonu (Performance), prostředí (Environment), aktuátorů (Actuators) a senzorů (Sensors) a označuje se akronymem PEAS. Při návrhu agentu je podle autorů nejprve potřeba co nejúplněji specifikovat úkolové prostředí.

Autoři dále uvádějí, že nezáleží na tom, zda se jedná o prostředí reálné, nebo umělé, podstatná je komplexnost vztahů mezi chováním agentu, vnímanou sekvencí generovanou prostředím a měřením výkonu. Prostředí se podle autorů dělí na plně pozorovatelné, částečně pozorovatelné a nepozorovatelné. Plně pozorovatelné prostředí se podle autorů vyznačuje tím, že senzory agentu jsou schopné kdykoliv umožnit přístup ke kompletnímu stavu prostředí, to je praktické, protože agent nepotřebuje ve své paměti udržovat stav prostředí. Prostředí může být podle autorů částečně pozorovatelné např. kvůli nedostatku senzorů, nebo kvůli nepřesnosti senzorů. Pokud agent nemá vůbec žádné senzory, pak se podle autorů jedná o nepozorovatelné prostředí.

Míra výkonu podle (Russell a Norvig 2016) vyhodnocuje chování agentu v prostředí. Autoři dále uvádějí, že racionální agent jedná tak, aby optimalizoval očekávanou hodnotu míry výkonosti vzhledem k sekvenci vjemů, které doposud přijal. Agent umístěný v prostředí podle autorů vygeneruje na základě vnímání posloupnost akcí. Autoři dále uvádějí, že pokud je tato sekvence akcí žádoucí, pak si agent vedl dobře. Tato představa o vhodnosti je podle autorů zachycena pomocí míry výkonnosti, která vyhodnocuje jakoukoli danou posloupnost stavů prostředí. Autoři dále upozorňují na nutnost hodnotit výkonnost podle stavu prostředí, nikoliv podle stavu agentů, protože agent může podlehnout sebeklamu. Podle autorů neexistuje žádné pevné měřítko výkonnosti pro všechny agenty a úkoly, vhodné měřítko vymyslí návrhář podle okolností.

Podle (Wooldridge 2009; Russell a Norvig 2016) lze výkonnost agentů určovat pomocí hodnotící funkce označované v originále jako *„Utility function“*, nebo *„Fitness function“*. Prospěšnost je podle autorů numerické vyjádření optimálnosti stavu prostředí, čím je vyšší, tím lépe. Úkolem agentu je podle autorů dosáhnout co nejvyšší prospěšnosti, přičemž není specifikováno, jak to má agent provést. Podle autorů lze úkol specifikovat jako funkci, která spojuje skutečnou hodnotu s každým stavem prostředí. Celkovou prospěšnost agentu lze podle autorů definovat několika způsoby. Podle autorů je například možné definovat prospěšnost agentu jako nejhorší stav, s nímž se agent může setkat. Další možností je podle autorů definovat celkovou prospěšnost jako průměr ze všech stavů, které mohou nastat. Autoři dále uvádějí, že neexistuje jediné dobré, nebo špatné řešení. Stanovení způsobu měření prospěšnosti závisí podle autorů na konkrétním úkolovém prostředí a druhu úkolu, který má agent provést. Hlavní nevýhoda tohoto přístupu spočívá podle autorů v tom, že přiřazuje prospěšnost jednotlivým stavům, proto je složité určit dlouhodobý vývoj. Jako řešení autoři navrhují specifikovat úkol jako funkci, která přiřadí prospěšnost ne jednotlivým stavům, ale celým postupům. Podle autorů je vhodné definovat hodnotící funkci tak, aby prospěšnost byla reálné číslo v normalizovaném tvaru v rozsahu od 0 (nejhorší hodnocení) do 1 (nejlepší hodnocení). Hodnotit prospěšnost lze například podle spotřeby zdrojů, přičemž mezi zdroje patří kromě materiálů a energií i čas (např. je vhodné minimalizovat dobu řešení úkolů a eliminovat nežádoucí prostoje agentů).

2.2.4 Strojové učení agentů

Agenty vybavené pamětí a schopností učení se při své činnosti mohou dále zdokonalovat. Jak uvádí (Russell a Norvig 2016) agent se učí, jestliže zlepšuje svůj výkon při řešení budoucích úkolů na základě pozorování prostředí. Učení má podle autorů mnoho podob, záleží na druhu agentu, komponenty, která má být zdokonalena a na dostupné zpětné vazbě. Každá komponenta agentu může být podle autorů zdokonalena pomocí učení z dat. Která technika má být použita pro zdokonalení závisí podle autorů na následujících faktorech:

- Která komponenta má být zdokonalena.
- Jaké má agent předchozí znalosti.
- Jaká reprezentace dat a komponenty je použita.
- Jaká zpětná vazba je dostupná pro učení.

Autoři dále uvádějí seznam komponent, které mohou být zdokonaleny:

1) Přímé mapování podmínek aktuálního stavu na akce.
2) Prostředek k odvození relevantních vlastností světa ze sekvence vjemů.
3) Informace o vývoji světa a o výsledcích akcí, které agent může uskutečnit.
4) Informace o hodnocení (prospěšnosti) naznačující stav světa.
5) Informace o užitečnosti akce označující vhodnost provedení akcí.
6) Cíle, které popisují kategorie stavů, jejichž úspěch maximalizuje užitečnost agentu.

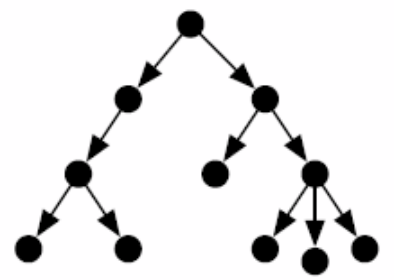
Podle (Russell a Norvig 2016) existují následující techniky strojového učení:

- Induktivní: Učení obecné funkce, nebo pravidla ze specifických párů vstup-výstup.
- Deduktivní: Přechod od známého obecného pravidla k novému pravidlu, které umožňuje účinnější zpracování.
- Učení bez učitele: Agent se učí vzory ve vstupních datech, ale nemá k dispozici zpětnou vazbu. Nejběžnější formou učení bez učitele je shlukování objektů (clustering), což je detekování potenciálně užitečných skupin vstupních příkladů.

- Posilovací učení: Agent se učí ze série posílení – odměn, nebo trestů. Na základě zpětné vazby agent rozhodne, které z akcí provedených před posílením k němu přispěly.
- Učení s učitelem: Agent pozoruje několik příkladů párů vstup-výstup a na základě zpětné vazby od učitele se naučí funkci, která mapuje od vstupu k výstupu. Učitelem může být i prostředí.
- Kombinované učení s učitelem a bez učitele: Agent dostane k dispozici několik označených příkladů (správné/špatné řešení), poté musí určit co nejvíce správných řešení z velké sbírky neoznačených příkladů.

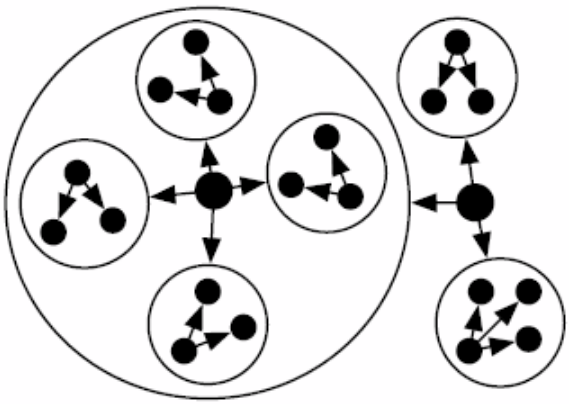
2.2.5 Organizační struktury agentů

Pro zlepšení výkonu MAS může být vhodné použít nějakou organizační strukturu. Stejně jako v lidských organizacích je podle (Horling a Lesser 2004) určeno jak na sebe členové populace vzájemně působí. Kromě toho nelze podle autorů určit jeden typ organizace vhodný pro všechny MAS. Autoři dále uvádějí, že organizace MAS je kolekce rolí, vztahů a struktur autority, které upravují chování systému. Uvnitř MAS může podle autorů koexistovat i více druhů organizačních struktur a mohou dynamicky vznikat a zanikat. Přehled klíčových vlastností, výhod i nevýhod nejběžněji používaných organizačních struktur podle (Horling a Lesser 2004) je uvedený v Tab. 1. Průmyslové aplikace typicky využívají hierarchické uspořádání, holarchie, nebo týmy.



Obr. 1: Hierarchická organizační struktura. Zdroj: (Horling a Lesser 2004).

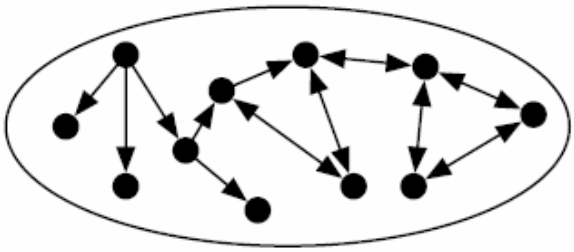
Hierarchie, nebo hierarchická organizace, viz Obr. 1, je podle (Horling a Lesser 2004) pravděpodobně nejstarší druh strukturovaného, organizačního návrhu aplikovaného v MAS. Agenty jsou podle autorů koncepčně uspořádány do stromové struktury, přičemž agenty umístěné ve stromu výše (blíže kořenu) mají globálnější pohled než agenty pod nimi, jedná se tedy o centralizovaný způsob řízení. Nevýhodou tohoto řešení podle autorů může být zahlcení výše ve stromu umístěných agentů, které však lze řešit klonováním těchto agentů. Další nevýhodou je podle autorů zpoždění rozhodnutí, které je dané nutností postupného procházení celé cesty ve stromu nahoru a zpět dolů. Toto zpoždění je podle autorů možné eliminovat použitím křížových odkazů umožňujících přímější komunikaci.



Obr. 2: Holarchická organizační struktura. Zdroj: (Horling a Lesser 2004).

Holarchie je podle (Horling a Lesser 2004) typem organizace, ve které se vše skládá z víceúrovňových skupinových hierarchií. Příkladem podle autorů mohou být biologické, astrologické a sociální systémy, např. vesmír se skládá z galaxií, ty jsou složené ze slunečních soustav atd. až k subatomárním částicím. Každá skupina holarchického uspořádání se nazývá holon a podle autorů může být zároveň celkem složeným z dalších holonů umístěných hierarchicky níže a zároveň částí holonu umístěného v hierarchii výše. Hierarchické vztahy jsou na Obr. 2 znázorněné orientovanými hranami, zatímco kruhy představují hranice holonů. Podle (Horling a Lesser 2004) je částečně autonomní holon charakteristickým rysem holarchie. Autoři dále uvádějí, že holony jsou obvykle vybavené dostatečnou autonomií k určení nejlepšího postupu plnění zadaných požadavků. Žadatel podle autorů

nemusí přesně vědět jakým způsobem bude jeho žádost vyřešena, díky tomu může být holon flexibilní a může koordinovat potenciálně se doplňující, nebo protichůdné úkoly. Tato vlastnost podle autorů snižuje zátěž kladenou na žadatele a umožňuje holonu dynamicky přizpůsobit své chování novým podmínkám, za předpokladu, že dojde ke splnění požadavku.



Obr. 3: Týmová organizační struktura. Zdroj: (Horling a Lesser 2004).

Tým agentů, viz Obr. 3, se podle (Horling a Lesser 2004) skládá z několika kooperujících agentů, které se dohodly na spolupráci při plnění společného cíle. Autoři dále uvádějí, že týmová organizační struktura má snahu upřednostňovat užitečnost týmu jako celku před jednotlivými členy. Podle autorů se předpokládá, že se agenty budou nějakým způsobem koordinovat tak, že jejich individuální akce budou v souladu s cílem týmu a povedou ke splnění cíle. Každý agent si podle autorů může převzít jednu, nebo více rolí potřebných k řešení dílčích úkolů požadovaných týmovým cílem. V průběhu plnění cíle se tyto role mohou podle autorů měnit v reakci na plánované, nebo neplánované události, zatímco samotný cíl týmu obvykle zůstává relativně konzistentní. Hlavní výhoda týmové práce podle autorů spočívá v tom, že skupina kooperujících agentů dokáže vyřešit náročnější úkoly, než by dokázal kterýkoliv člen týmu samostatně. Mezi výhody týmu podle autorů patří ještě redundance a schopnost splnit globální omezení. Členové týmu jsou podle autorů schopné explicitně uvažovat o důsledcích interakcí mezi agenty, což dává týmu potřebnou flexibilitu pro práci v nejistém prostředí s nepředvídatelnými podmínkami.

Paradigma (název v angličtině)	**Klíčové vlastnosti**	**Výhody**	**Nevýhody**
Hierarchie (Hierarchy)	Dekompozice	Použitelnost pro mnoho běžných domén, velmi dobrá škálovatelnost	Křehkost, mohou vznikat úzká hrdla, nebo zpoždění
Holarchie (Holarchy)	Dekompozice s autonomií	Autonomie funkčních jednotek	Nutnost organizovat holony, nedostatečná předvídatelnost výkonu
Koalice (Coalition)	Dynamika, zaměření na cíl	Využití síly v číslech	Krátkodobé výhody nemusí převažovat stavební náklady organizace
Tým (Team)	Soudržnost skupiny	Zaměření na větší problémy, zaměření na úkoly	Vyšší míra komunikace
Kongregace (Congregation)	Dlouhodobost, zaměření na užitek	Usnadnění vyhledávání agentů	Sady mohou být příliš restriktivní
Společenství (Society)	Otevřenost systému	Veřejné služby, dobře definované úmluvy	Potenciální složitost, agenty mohou vyžadovat další společenské schopnosti
Federace (Federation)	Zprostředkující agenty	Zprostředkování, překladatelské služby, usnadňuje práci s fondem dynamických agentů	Zprostředkovatelé se stávají úzkým hrdlem
Trh (Market)	Konkurence prostřednictvím cen	Dobré alokace, vyšší užitečnost díky centralizaci, vyšší férovost nabídek	Může dojít k nežádoucímu chování z důvodu tajných dohod, může dojít k přidělení rozhodnutí s vysokou složitostí
Matice (Matrix)	Více manažerů	Sdílení zdrojů, mnohonásobné ovlivnění agentů	Může dojít ke konfliktům, je nutná zvýšená propracovanost agentů
Kombinace (Compound)	Kombinace organizace	Využití výhod několika typů organizací	Zvýšená složitost, nevýhody několika typů organizací

Tab. 1: Organizační struktury MAS. Zdroj: (Horling a Lesser 2004).

2.3 Výrobní logistika

Výrobní logistika (též produkční logistika, PL) odpovídá za transport a skladování veškerých surovin, materiálů, polotovarů i hotových výrobků. Její optimalizace tudíž může výrazně ovlivnit celkovou produktivitu zpracovatelského řetězce.

Podle (Tucnik et al. 2017) je obecně doporučováno, aby byl hlavní sklad i haly s výrobními linkami v prostoru továrny. Autoři dále uvádějí, že každá výrobní linka má k dispozici úložný prostor s omezenou kapacitou a řešit je tedy nutné zejména prostorová omezení. Velikost úložného prostoru se podle autorů může u jednotlivých výrobních linek lišit.

V práci (Tucnik et al. 2017) je uvažován deterministický požadavek na doručení surovin pro každou z výrobních linek, ten lze odvodit z plánu výroby. Z důvodu omezení velikosti hlavního skladu a úložných prostorů linky, viz výše, musí být podle autorů každý požadavek na suroviny přepočítán na objemové jednotky. Kapacity hlavního skladu a úložných prostor jsou podle autorů určeny událostmi v průběhu času a průběžně se mění. Množství konkrétních komponent v úložišti je podle autorů kontrolováno ve chvíli, kdy finální výrobek opouští výrobní prostor. Autoři dále uvádějí, že pokud množství komponent v úložišti postačuje pro výrobu posledního plánovaného výrobku, tak výrobní linka začne objednávat materiál z hlavního skladu. Objednávky jsou podle autorů tvořeny v poměru dostatečném pro výrobu kompletních výrobků ve vztahu ke zbývající době trvání směny a omezeným kapacitám úložišť.

Podle (Lu et al. 2019) jsou během reálného provozu dopravní prostředky náchylné blokování, až zablokování, což znesnadňuje plánování produkční logistiky. Autor dále uvádí, že cílem vědců a podniků je lepší organizace PL.

Podle (Blesing et al. 2017) v automobilovém průmyslu probíhá nepřetržitá hromadná výroba s využitím vysoce účinných výrobních a montážních linek. Autor dále upozorňuje na nutnost velkého úsilí v procesech plánování, aby byly výrobní linky plně využity. Důvodem je podle (Blesing et al. 2017) vysoká složitost struktury produktů s velkou rozmanitostí variant a zároveň vysoký počet dílů, které musí být dodány včas a na správné místo výrobní linky. Autor dále představuje projekt

SMART FACE, který se zaměřuje na vývoj simulačního a prototypovacího demonstrátoru, použitého v systému plánování a kontroly výroby na bázi agentů pro pohyb karosérií a dílů aut na flexibilní montážní lince. Výhodou tohoto projektu je podle autora, že zpoždění v jednom kroku nevede ke zpoždění jiných objednávek a může být kompenzováno jinými stanicemi na lince. V práci (Blesing et al. 2017) jsou použity algoritmy pro přiřazení úkolů a plánování cest automatizovaných řízených vozidel (Automated Guided Vehicle, AGV). Podle autora je systém výběru vhodného AGV pro provedení konkrétního úkolu řešen FIPA aukcemi, kde AGV vystupují jako dražitelé a stanice jako prodejci. Nabídky v aukcích byly podle autora stanoveny na základě obsazenosti a polohy vozidla a ukázalo se, že decentralizovaný systém je oproti centrálnímu plánování efektivnější, vozidla byla využívána více a zároveň se zkrátila doba zpracování objednávek. V práci (Blesing et al. 2017) je použitá implementace algoritmu (ter Mors 2011) s časovými sloty pro výpočet nekonfliktních plánů tras s ohledem na stávající plány jiných vozidel.

V práci (Luo et al. 2017) je použitý pojem synchronizovaná výroba a logistika (Synchronized Production and Logistics, SPL) a také framework Ubi-SPL (Synchronized Production and Logistics via Ubiquitous Technology), který byl úspěšně realizován jako případová studie v chemickém průmyslu. Autor na základě výsledků implementace uvádí, že díky použití synchronizovaného rozhodovacího algoritmu došlo ke snížení průměrné doby čekání na paletu produktu o 19 %. Díky zkrácení doby vyhledání palet a doby nalezení skladové zóny došlo podle autora ke snížení průměrné doby vnitřní přepravy o 38 %. Celková průměrná doba výroby a logistiky se díky implementaci systému Ubi-SPL podle autora zlepšila o 20 %. Autor ve své práci uvádí následující závěry:

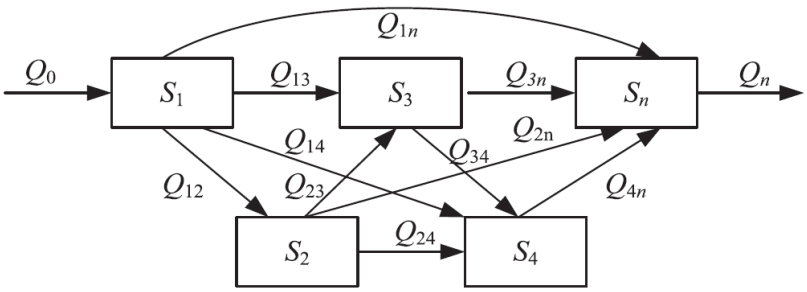
1) Synchronizovaný postup rozhodování vyvážil požadavky mezi výrobním a skladovým oddělením.
2) Hladký chod skladu a vysoké využití prostoru zvyšují efektivitu výroby.
3) Sběr, zpracování a systematické rozhodování v reálném čase snižují nejistotu a dynamiku lidských faktorů.

V dynamických výrobních systémech není podle (Zhang a David 2019) logistika pouze předpokladem nepřetržitého provozu výrobního systému, ale také

procesem zvyšování hodnoty. Výběr výrobního modelu a míru využití kapacity ve výrobním systému by podle autorů mohla výrazně ovlivnit úroveň provozu materiálů, zařízení, personálu, společně s dalšími faktory. Autoři dále uvádějí, že při vyhodnocování systémů výrobní logistiky používají některé výzkumné metody grafickou analýzu, nebo kvalitativní popis, proto je složité přesně vypočítat efektivitu provozu výrobní logistiky. Některé studie podle autorů sice přijaly některé z kvantitativních postupů, ale jejich výzkumné cíle nebyly zaměřeny na efektivitu provozu výrobní logistiky.

Podle (Zhang a David 2019) lze rozdělit výrobně logistické systémy z pohledu jejich složek do pěti klíčových faktorů, viz Obr. 4. Autoři ve své práci popisují jednotlivé faktory takto:

- Výrobek: Výrobek odkazuje na hmotnou podstatu a je zároveň hlavním tělesem toku v produkčně logistickém systému, na Obr. 4 označeno Q.
- Transport: Faktor transportu je definován jako zařízení, nebo zpracovatelské vybavení, které podporuje tok produktu, na Obr. 4 označeno S.
- Směr toku: Směr toku určuje směr toku od výchozího bodu k finálnímu výrobku, na Obr. 4 označeno směrem šipky.
- Propustnost: Propustnost představuje množství produktu přepravovaného pomocí transportu v určitém směru toku, tedy hodnotu faktoru výrobku Q.
- Proces: Proces je definován jako počet cest různých transportů. Představuje počet cest pro cesty určitého směru toku pro produkt, který prochází transportem. Např. proces $S_1S_3 = Q_{13} + Q_{12} + Q_{23}$ na Obr. 4



Obr. 4: Souvztažnost klíčových faktorů v produkčně logistickém systému. Zdroj: (Zhang a David 2019).

Tyto klíčové faktory jsou podle autorů definovány a poté integrovány do vyhodnocovacího systému. Indexy, které mají velký vliv na různé klíčové faktory, včetně produktu, transportu, směru toku, propustnosti a procesu, jsou vybírány tak, aby vytvořily systém hodnocení provozu logistiky výroby.

2.4 Dodavatelský řetězec

Dodavatelský řetězec je podle (Zhao a Zhao 2018; Li et al. 2019; Long 2015; Prasad et al. 2017), sítí produktů, informací a služeb spojených se schopnostmi nabídky, výroby, distribuce a poptávky. Funkční model struktury síťového řetězce podle autorů spojuje dodavatele, výrobce, distributory a koncové uživatele do jednoho celku. Jak autoři dále uvádějí, podstatou dodavatelského řetězce je plánování, organizace, koordinace a řízení procesů s přidanou hodnotou při nákupu, zpracování, balení a přepravě od surovin po finální výrobky.

Továrna má být podle (Tucnik et al. 2017) schopná reagovat a vyjednávat s dodavateli o poptávkách. Pro rozhodnutí, zda uzavřít, nebo neuzavřít smlouvu je podle autorů důležité vědět, jestli jsou termíny doručení dosažitelné. Autoři dále uvádějí, že vedení továrny rovněž potřebuje znát nejkratší možný čas, za který je dodavatel schopný splnit očekávané požadavky. Proto je podle autorů výpočet doby výroby finálního výrobku důležitou charakteristikou. Na každé z výrobních linek jsou podle autorů rozeznávány tři typy časově náročných činností. Těmi jsou podle autorů celkový čas potřebný pro přípravu komponent, vlastní doba výroby a doba balení výrobku.

Manažeři dodavatelského řetězce podle (Prasad et al. 2017) usilují o snížení nákladů dodavatelského řetězce, které obvykle sestávají z nákladů souvisejících s výrobou, nákladů souvisejících se zásobami a nákladů spojených s distribucí. Obecně se však výrobní manažeři podle autorů soustředí pouze na svou doménu, tj. snižují náklady spojené s výrobou a zásobami, distribuci zadávají externím poskytovatelům logistiky. Jak autoři dále uvádějí, řízení dodavatelského řetězce je aktivní oblastí výzkumu v posledních dvou desetiletích. Podle autorů se ukázalo, že optimalizace celého dodavatelského řetězce je důležitější než optimalizace jeho částí, vyžaduje však maximální spolupráci ze strany složek dodavatelského řetězce.

Konkrétně autoři zdůrazňují nutnost sdílení přesných informací (např. o výrobních nákladech, distribučních nákladech atd.) mezi všemi složkami dodavatelského řetězce (vedoucími závodů, poskytovateli distribuce atd).

Podle (Zhao a Zhao 2018) lze dodavatelské řetězce rozdělit na základě stability, kapacity dodavatelského řetězce a funkčních modelů dodavatelského řetězce do těchto tří kategorií:

1) Stabilní/dynamický dodavatelský řetězec: V různých odvětvích se podle (Zhao a Zhao 2018) vyskytuje různá složitost dodavatelského řetězce. Například dodavatelský řetězec ovoce, zeleniny a dalších průmyslových odvětví je podle autorů relativně jednoduchý, snadno se koordinuje a spravuje, proto se nazývá stabilní. Autoři dále uvádějí, že v automobilovém průmyslu má dodavatelský řetězec relativně složitou strukturu sítě. Podle (Zhao a Zhao 2018; Guo et al. 2013) se taková struktura může skládat ze stovek dodavatelů, distributorů a výrobců. Stabilita tohoto dodavatelského řetězce je podle autorů nízká, proto je vyžadováno dobré řízení a organizace.
2) Vyvážený/nevyvážený dodavatelský řetězec: V současné době se podle (Zhao a Zhao 2018) výroba přizpůsobuje poptávce zákazníků, změny poptávky jsou však jen obtížně zachytitelné a jsou ovlivněny mnoha faktory. Pokud je výrobní kapacita dodavatelů a výrobců v dodavatelském řetězci v souladu s potřebami zákazníků, je podle autorů celý dodavatelský řetězec ve stavu rovnováhy. Pokud se však poptávka trhu výrazně změní a dodavatelé a výrobci nemohou na tyto změny rychle reagovat, může se podle autorů dodavatelský řetězec změnit na nevyvážený. Proto je v procesu řízení dodavatelského řetězce nezbytné vyhodnocovat a analyzovat schopnost rychlé reakce dodavatelského řetězce, případně přijímat různé strategie pro dosažení rovnováhy.
3) Efektivní/reaktivní dodavatelský řetězec: Podle (Zhao a Zhao 2018) existuje několik modelů řízení dodavatelského řetězce. Efektivní dodavatelský řetězec je podle autorů obvykle provozovaný s nejnižšími náklady. Dodavatelský řetězec, který rychle reaguje na změny poptávky a trhu se podle (Zhao a Zhao 2018; Chen a Song 2014) nazývá reaktivní.

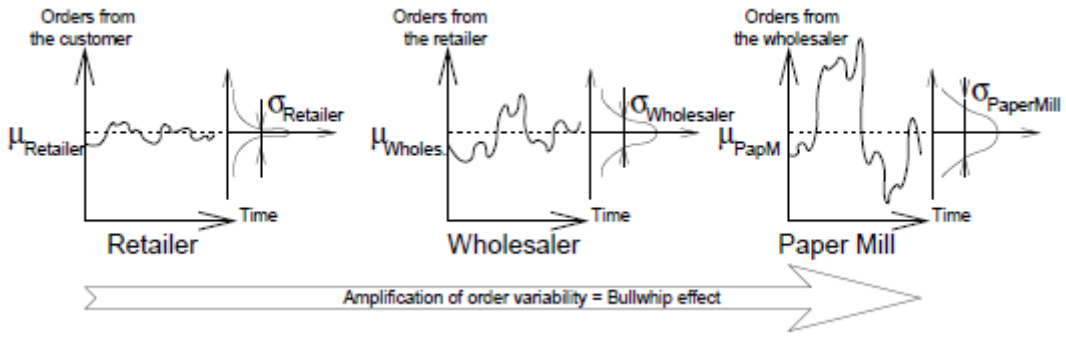
Při optimalizaci dodavatelského řetězce je podle (Zhao a Zhao 2018) nutné zvážit všechny výše uvedené vlastnosti dodavatelského řetězce.

V práci (Li et al. 2019) autoři sledují vliv počtu výrobních partnerů na model evoluční hry technologie výroby. Podle autorů lze rozdělit výrobce v dodavatelském řetězci v závislosti na počtu partnerů na výrobce s více partnery a výrobce s méně partnery. Během této hry podle autorů obvykle mohou obě strany přijmout strategii kolaborace, nebo dominance. Jak autoři dále uvádějí, v rámci strategie kolaborace výrobci počítají se zaplacením výrobní technologie a nákladů na výměnu zařízení za účelem dosažení spolupráce. V rámci dominantní strategie výrobci podle autorů očekávají, že budou formulovat výrobní technické normy a po druhé straně vyžadují nahrazení výrobní technologie a zařízení podle vlastních technických norem. Pokud obě strany přijmou strategii kolaborace, počet nahraditelných partnerů obou stran bude mít podle autorů významný vliv na stav vzájemného vyjednávání a zároveň stanoví dvoustranný úkol na odstranění rozdílů v technologii výroby. Jak autoři dále uvádějí, pouze v tomto případě existuje nadstandardní zisk způsobený účinky dvoustranné koordinace. Pokud jedna strana přijme kolaborační strategii a druhá strana přijme dominantní strategii, tak podle autorů ponese celkové náklady na odstranění rozdílů v technologii výroby pouze strana, která přijala strategii kolaborace. Pokud však obě strany zaujmou dominantní strategii, tak podle autorů nelze vytvořit kooperativní výrobní vztah. Jak autoři dále uvádějí, když dva výrobci spolupracují na výrobě nového produktu, nebo součásti a jedna strana má více nahraditelných partnerů, tak je její vyjednávací síla ve hře nákladů na odstranění rozdílů technologií vyšší. To však podle (Li et al. 2019; Taurino a Villa 2017) neznamená, že by výrobci měli mít stále více výrobních partnerů, místo toho by měli udržovat počet partnerů v rozumném rozsahu. Důvod je ten, že pokud existuje větší rozdíl mezi počtem výrobních partnerů obou stran, pak by strana, která má méně výrobních partnerů, byla ve hře standardů technologie výroby v nevýhodném stavu. Tato úvaha podle autorů naznačuje, že výše uvedený formální model založený na teorii her může být použit pro vyjasnění konceptu multiagentové sítě, ale musí jej následovat ověření výsledků vyjednávání a účinnosti spolupráce agentů.

V decentralizovaných modelech řízení dodavatelského řetězce lze podle (Prasad et al. 2017) použít inteligentní agenty. Podle autorů lze, každého, kdo v

dodavatelském řetězci činí rozhodnutí modelovat jako agent. Například pro použití v decentralizovaném systému plánování distribuce výroby se dvěma agenty lze podle autorů navrhnout produkční agent a distribuční agent. Kromě toho může podle autorů být přidán také plánovací agent, pokud správce požaduje plánování činnosti dodávky, výroby a distribuce. Jak autoři dále uvádějí, komunikaci mezi výše zmíněnými agenty může zajistit komunikační agent. Každý agent má podle autorů svůj cíl (např. distribuční agent usiluje o minimalizaci distribučních nákladů) a soubor omezení (např. distribuční agent nesmí překročit kapacitu dopravce). Následně podle autorů všechny spolupracující agenty vytvoří plány výroby a distribuce, které budou přijatelné pro všechny.

Jeden z problémů průmyslu a dodavatelských řetězců se nazývá „Efekt biče“ (v angličtině Bullwhip effect). Tento efekt podle (Chaib-draa a Müller 2006; Carlsson a Fuller 2000) spočívá ve zvýšení variability objednávky. Autoři dále uvádějí, že zvýšení variability je problém, protože dochází k nepředvídatelnému zvýšení poptávky, a tedy i objednávky. Výše uvedený problém autoři demonstrují na příkladu jednoduchého dodavatelského řetězce tvořeného maloobchodem, velkoobchodem a výrobcem (papírnou). Vzory objednávek těchto tří společností jsou podle autorů podobné, variabilita objednávek se však postupně zvyšuje ve směru od maloobchodu k výrobci. Míra variability efektu biče je uváděna jako směrodatná odchylka objednávek σ, viz Obr. 5.



Obr. 5: Efekt biče v dodavatelském řetězci. Zdroj: (Chaib-draa a Müller 2006).

Efekt biče sám o sobě není podle autorů škodlivý, jeho důsledky však ano. Autoři dále uvádějí výčet důsledků efektu biče způsobujících zvýšené náklady:

1) Nadměrné investice do zásob: Efekt biče zvyšuje nepředvídatelnost poptávky, proto je třeba, aby se všechny společnosti v dodavatelském řetězci zabezpečily proti odchylkám a vyhnuly se tak nedostatku zásob.
2) Zhoršení úrovně zákaznických služeb: I přes nadměrné úrovně zásob může nepředvídatelnost poptávky způsobit nedostatek skladových zásob.
3) Ztráta příjmů: Kromě zhoršení úrovně zákaznických služeb může nedostatek zásob také způsobit ztrátu příjmů.
4) Snížení produktivity: Z důvodu ztráty příjmů je provoz méně efektivní i v oblasti nákladů.
5) Obtížnější rozhodování: Tvůrci rozhodnutí reagují na fluktuace poptávky a přizpůsobují výrobní a skladové kapacity tak, aby vyhověli požadavkům ve špičce.
6) Neefektivní přeprava: Plánování přepravy je ztíženo kvůli nejistotám poptávky.
7) Neefektivní výroba: Podobně jako v případě dopravy je plánování výroby ztíženo nejistotami poptávky, což může způsobit zmeškání výrobních plánů.

(Chaib-draa a Müller 2006; Carlsson a Fuller 2000) dále uvádějí, že tyto důsledky nejsou způsobeny změnami poptávky koncových zákazníků, ale pouze neefektivitou v dodavatelském řetězci. Vznik efektu biče může mít podle autorů několik příčin: může se jednat o nepřesné metody předpovědí poptávky, nevhodné strategie společností (poptávka převyšuje nabídku), hromadění objednávek, případně změny cen produktů, které podněcují klienty, aby nakupovali při nízké ceně. Řešením efektu biče by podle autorů mělo být především sdílení informací o poptávce. Takové sdílení informací je však podle autorů možné pouze v případě, že společnosti spolupracují.

Podle (Chaib-draa a Müller 2006) lze v literatuře nalézt argumenty pro i proti využívání multiagentových systémů v řízení dodavatelského řetězce. (Parunak 1996) porovnává řízení dodavatelského řetězce pomocí autonomních agentů s řízením pomocí konvenčních technologií. Za tímto účelem autor identifikuje multiagentní systémy jako biologické (ekosystémové) a ekonomické (tržní) modely,

zatímco tradiční přístupy jsou porovnávány s vojenskými vzory hierarchické organizace. Hlavní nevýhody MAS podle autorů spočívají v:

1) Teoretické optimum nelze zaručit, protože neexistuje celkový pohled na systém.
2) Předpovědi pro autonomní agenty lze obvykle provádět pouze na agregované úrovni.
3) Výpočetní systémy autonomních agentů se mohou stát nestabilními, protože podle systémové dynamiky je jakýkoli systém potenciálně nestabilní.

Autonomní přístup založený na agentech však má podle autorů také následující výhody:

1) Každý agent je blízko místa kontaktu se skutečným světem, takže výpočetní stavy systému umožňují velmi pečlivě sledovat stav světa.
2) V MAS není nutná centralizovaná databáze.
3) Celkové chování systému vychází z lokálních rozhodnutí, proto se systém dokáže automaticky přizpůsobit šumu v prostředí.
4) Díky tomu také lze libovolně přidávat, nebo odebírat agenty.
5) Software každého agentu je mnohem kratší a jednodušší, než jaký by byl vyžadován centralizovaným přístupem. To usnadňuje psaní, ladění a údržbu kódu.
6) Systém při svém běhu sám plánuje. Neexistuje žádná samostatná fáze plánování, tudíž není třeba čekat na dokončení plánovače. Optima vypočítaná konvenčními systémy navíc nemusí být realizovatelná v praxi, a podrobnější předpovědi získané konvenčními přístupy jsou skutečným světem často znehodnoceny.

Podle (Chaib-draa a Müller 2006) všechny tyto důvody ukazují význam pro použití agentů v řízení dodavatelského řetězce. Jak autoři dále uvádějí, systémy založené na agentech jsou díky své přizpůsobivosti, autonomii a sociálním schopnostem životaschopnou technologií pro implementaci komunikace a rozhodování v reálném čase.

Podle (Chaib-draa a Müller 2006) znemožňuje složitá interakce mezi entitami a vícevrstvou strukturou dodavatelských řetězců použití analytických modelů, které mohou přesně zachytit dynamiku celých dodavatelských řetězců. Systémy založené na agentech jsou podle autorů slibnou alternativou pro modelování a simulaci dodavatelského řetězce. Autoři především zdůrazňují využití následujících vlastností agentů: reaktivita, sociální schopnosti a proaktivita. Díky těmto vlastnostem jsou podle autorů agenty ideální pro modelování a analýzu dodavatelských řetězců, kde je nezbytná spolupráce, inteligence a mobilita. Kromě toho autoři, u technologií založených na agentech, uvádějí i možnost využití podpory souběžného a distribuovaného rozhodování, které je nedílnou součástí řízení dodavatelského řetězce. Přístupy založené na agentech jsou podle autorů všestranné, a kromě kvantitativních aspektů, pro které jsou nejvhodnější tradiční přístupy modelování, mohou snadno zachycovat kvalitativní a transakční události v dodavatelském řetězci.

2.5 Konfigurace a plánování pracovních postupů

Konfigurace a plánování kolaborativních pracovních postupů je podle (Hsieh 2019) náročný úkol, protože zahrnuje výpočetní složitost, distribuovanou architekturu a závislost mezi pracovními postupy různých partnerů. Pro dosažení flexibility a snížení nákladů a času souvisejících s konfigurací sítě dodavatelského řetězce, autor navrhuje přístup, který kombinuje MAS, kontraktační síťový protokol, modely pracovních postupů a automatickou transformaci modelů pracovních postupů tak, aby se dynamicky formuloval problém s plánováním. Pro dosažení škálovatelnosti, navrhl autor algoritmus, řešící problém optimalizace pomocí kolaborativního a distribuovaného výpočetního schématu. Jak autor dále uvádí, pro dosažení flexibility, vč. možností rekonfigurace a škálování musí být splněno několik požadavků:

1) Pracovní postup společnosti v dodavatelských řetězcích musí být popsán a specifikován ve standardním formátu.
2) Platforma MAS použitá pro implementaci musí podporovat informační infrastrukturu a interakční protokoly definované organizací průmyslových standardů, aby bylo dosaženo interoperability mezi agenty.
3) Algoritmus vytváření rozvrhu musí být vyvinut na základě dynamického publikování a objevování služeb.
4) Metoda plánování musí být škálovatelná. Toho je dosaženo použitím distribuované výpočetní architektury založené na strategii rozděl a panuj.
5) Pro dosažení interoperability musí být všechny informace v procesu vyjednávání, včetně výzvy k předkládání návrhů, vlastních návrhů, zadávání zakázek a uzavírání kontraktů, popsány na základě standardního formátu, např. XML.

Podle (Hsieh 2019) existují v protokolu kontraktační sítě dvě role, ve kterých může agent vystupovat: manažer, nebo dražitel. Zahájení smlouvy mezi manažerem a jedním, či více dražiteli zahrnuje podle autora následující čtyři fáze:

1) Výzva k předkládání návrhů (“Call For Proposals”, zkráceně CFP): Manažer oznámí úkol všem potenciálním dražitelům. Oznámení obsahuje popis úkolu.
2) Předkládání návrhů: Po obdržení oznámení o nabídkovém řízení vypracují dražitelé, kteří jsou schopni provést úkol, návrhy a předloží je manažerovi.
3) Zadání zakázky: Po přijetí a vyhodnocení předložených návrhů manažer udělí zakázku nejlepšímu dražiteli.
4) Uzavření smlouvy: Vybraný dražitel se může buď zavázat k provedení úkolu, nebo odmítnout přijmout kontrakt zasláním zpráv manažerovi. V případě odmítnutí manažer znovu vyhodnotí nabídky a zadá zakázku/zakázky jinému uchazeči/uchazečům.

2.6 Trasový algoritmus a rozhodování

Trasový algoritmus musí podle (Blesing et al. 2017; ter Mors 2011) zajistit, aby každý dopravník našel nejkratší trasu od startu do cílového umístění, nesmí však dojít ke střetu s kterýmkoli jiným agentem a nesmí dojít k vzájemnému zablokování. Za tímto účelem byl podle autorů implementován grafový směrovací algoritmus, založený na principu kontextového plánování tras (*„Context Aware Route Planning“*, zkráceně CARP). Vypočítané trasy lze podle autorů definovat takto:

$$\pi_n = ([r_1, \tau_1], \dots, [r_n, \tau_n], \tau_i = [t_i, t'_i]) \quad (1)$$

π_n Graf zdrojů

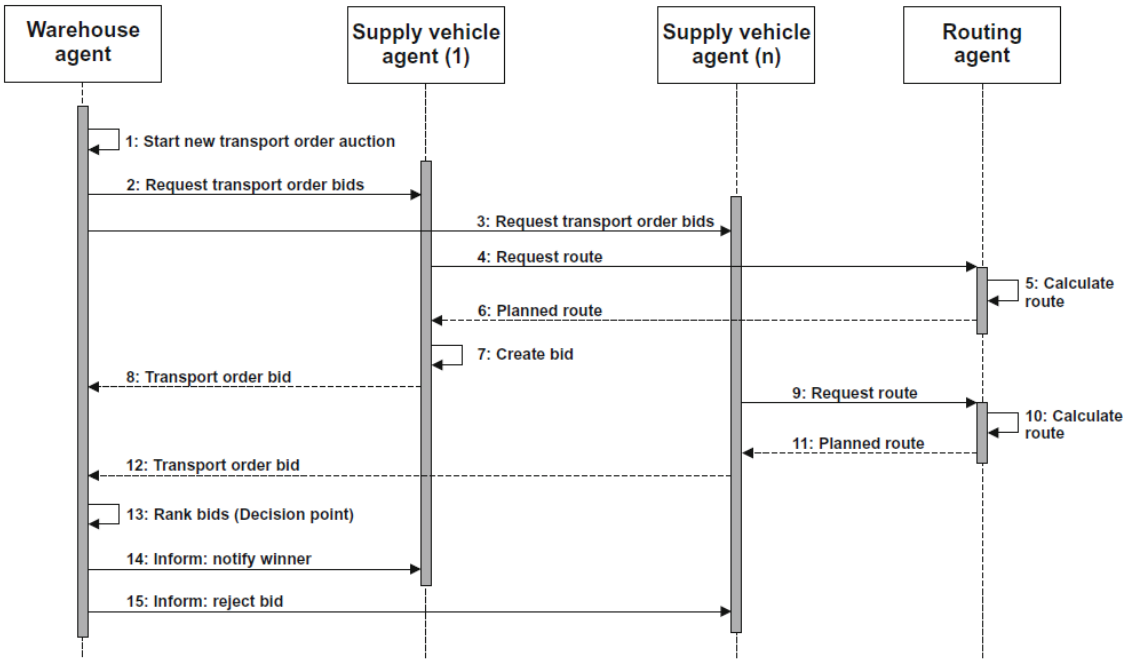
r_n Uzel grafu zdrojů

τ_n Časové okno dostupnosti (prostředek r_n je dostupný od času t_i do času t'_i)

Základní myšlenku CARP lze podle (Blesing et al. 2017; ter Mors 2011) definovat tak, že v rámci směrování plánují jednotlivé agenty své cesty postupně jeden po druhém. Pokud agent n plánuje svou trasu, jsou již naplánované (n-1) trasy podle autorů zahrnuty do aktuálního výpočtu. Jak autoři dále uvádějí, každý uzel grafu prostředků má vlastní časová okna, která definují, kdy je uzel volný. Pro plánování tras bez konfliktů jsou podle autorů vyžadována překrývající se časová okna mezi jednotlivými uzly trasy. Výpočet potřebných časových oken se podle autorů provádí na základě reálného času dopravníků. Jak autoři dále uvádějí, tyto časy lze použít k odhadu polohy vozidla v daném časovém bodě. Na základě této informace lze podle autorů vytvořit časová okna každého uzlu v grafu, kromě toho algoritmus poskytuje možnost začlenit do plánování budoucích tras aktuální události (například zpoždění).

Pro rozhodování na základě různých vstupních proměnných lze podle (Blesing et al. 2017) použít různé přístupy, např. genetické algoritmy, fuzzy logiku, nebo neuronové sítě. Nevýhodou genetických algoritmů a neuronových sítí je podle autorů vysoká složitost, proces rozhodování navíc v tomto případě není srozumitelný. Naopak velkou výhodou použití fuzzy logiky je podle autorů to, že proces řešení lze převést na operace pochopitelné lidmi. Podle (Blesing et al. 2017;

Branisso et al. 2013) byly nejlepší výsledky, při použití různých technik zvyšujících výkon flotily AGV ve skladu, dosaženy použitím fuzzy logiky v rozhodovacím procesu. (Blesing et al. 2017) dále uvádí, že součástí nabídky, která byla zaslána agentem zásobovacího dopravníku skladovému agentu během aukce přepravních objednávek, jsou tři hodnoty. Podle autorů se jedná o cestovní vzdálenost mezi zdrojovým a cílovým místem, dobu jízdy a stav nabití baterie vozidla. Jak autoři dále uvádějí, na základě těchto tří vstupních hodnot byly vytvořeny odpovídající fuzzy sady. Výstupní sada určuje podle autorů hodnotící funkci pro každé CTV, které podalo nabídku. Následně podle autorů skladový agent oznámí výherce, viz Obr. 6.



Obr. 6: Sekvenční diagram dražby transportních zakázek. Zdroj: (Blesing et al. 2017).

2.7 Modelování a ověřování

S rostoucí složitostí ekonomických systémů podle (Kamiński a Szufel 2013) dochází k tomu, že standardní matematické techniky modelování přestávají být dostatečné. Autoři dále uvádějí, že simulační techniky se v operačním výzkumu objevily kolem roku 1950, první modelovací nástroje založené na agentech však byly vyvinuty až po roce 1990. Agentově orientovaný přístup k ekonomickému modelování je podle autorů založený na pozorování, že chování ekonomického

systému v makro měřítku je výsledkem interakčního chování v mikro měřítku. Tento princip se označuje termínem emergence. Podle (Kubík 2004) *„není jasné co se přesně myslí mikro a makro úrovní, chováním, vlastnostmi, prvky, jaké podmínky musí být splněny, aby k danému chování došlo, z jaké perspektivy lze o systému usuzovat, atd.“*.

(Tucnik et al. 2017) ve své práci uvádí, že agenty mají širokou škálu charakteristik společných s ekonomickými subjekty, ať už se společnostmi, nebo jednotlivci. Mezi tyto charakteristiky podle autora patří především schopnost samostatného rozhodování a zaměření na splnění cílů, proto agenty mají potenciál činit racionální rozhodnutí. Autor dále vyzdvihuje následující schopnosti agentů: vnímání prostředí, určování racionálního postupu akcí a přizpůsobivost změnám prostředí. Takové schopnosti jsou podle autora velmi užitečné v dynamických prostředích, což je společná charakteristika většiny ekonomických systémů a modelů. Fungování ekonomiky je podle autora vždy spojené s řízením dodavatelského řetězce, bez ohledu na to, zda se jedná o skutečný, nebo virtuální svět.

2.8 Hodnocení efektivity výrobního procesu

Jak uvádí (Plinere a Aleksejeva 2019) zlepšení efektivity výroby může být dosaženo různými způsoby, např. pomocí efektivního vytížení výrobních zařízení, zkrácením času výroby, snížením výrobních nákladů a zavedením hubené výroby. Podle (Luo et al. 2017) lze optimalizace dosáhnout pomocí zlepšení spolupráce jednotlivých segmentů výrobního závodu. Jak autoři dále uvádějí, zejména výrobu a logistiku je možné transformovat ze samostatných a nezávislých činností na koordinovaný a synchronizovaný způsob spolupráce. (Plinere a Aleksejeva 2019) ve své práci informují o použití agentu plánování výroby (production scheduling agent). Podle autorů zahrnuje plánování výroby a zlepšení výrobního procesu tyto dílčí úkoly: efektivní pracovní vytížení výrobních zařízení, zkrácení dodací lhůty a organizování nepřetržité montáže výrobku. Cílem plánovače výroby je podle autorů organizovat výrobní proces tak, aby došlo ke zkrácení výrobní doby, ale zároveň byla zachována, nebo zvýšena úroveň kvality produktů. Jak uvádějí (Plinere a

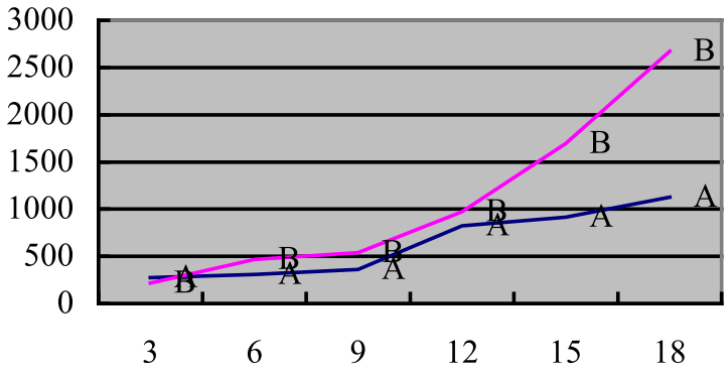
Aleksejeva 2019; Plinere a Borisov 2015) v systému výroby je možné použít agent řízení zásobování, jehož úkolem je efektivní řízení zásobování zahrnující předpověď budoucích požadavků, stanovení informací o doplňování a řízení úrovně zásob. Systémy řízení zásobování podle autorů mohou poskytovat specifické procesy, kterými jsou:

- Klasifikace ABC umožňuje přiřazení priorit výrobnímu času a finančním zdrojům
- Algoritmy pro předpověď budoucích požadavků a měření výkonu
- Stanovení informací o množství a frekvenci objednávek

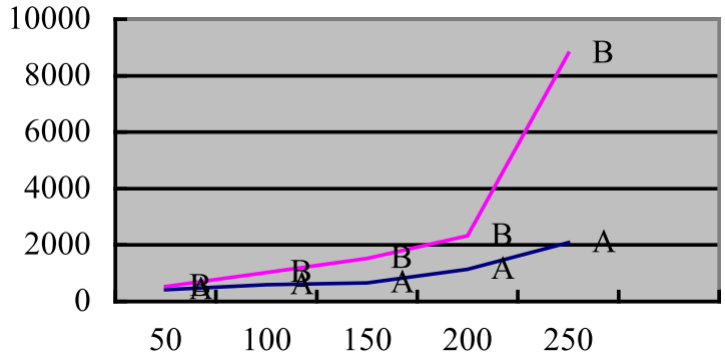
Agenty plánování výroby používají podle autorů pro plnění svých cílů výsledky činnosti agentu řízení zásobování.

Měření výkonu dodavatelského řetězce může být podle (Plinere a Aleksejeva 2019) klasifikováno do dvou rozsáhlých kategorií. Jednou z nich je podle autorů měření kvality vzhledem k uspokojení zákazníka, zahrnující rovněž tok informací a materiálu a efektivní řízení rizik. Druhou kategorií je podle autorů měření kvantity vzhledem k minimalizaci nákladů, maximalizaci zisku, maximalizaci míry plnění, minimalizaci doby odezvy zákazníka, spolehlivosti dodavatelů a minimalizaci dodací lhůty. Pro zvýšení účinnosti výroby je podle autorů velmi důležitá doba výroby, procento vadných výrobků, využití kapacity výrobního zařízení bez zbytečného čekání a úroveň zákaznických služeb.

K ověření analýzy popisované v práci (Hsieh 2019) autor experimentálně porovnal řešení založené na MAS (na obrázcích znázorněno křivkou A) s řešením založeným na centralizovaném řízení CPLEX firmy IBM (IBM 2015) (na obrázcích znázorněno křivkou B). Při rostoucí hloubce dodavatelských řetězců byla podle autora doba odezvy MAS podstatně kratší než doba odezvy centralizovaného řešení problémů, viz Obr. 7. Výrazně kratší doba odezvy MAS je podle autora patrná rovněž při zvyšování poptávky, viz Obr. 8. U obou grafů reprezentuje svislá osa dobu odezvy v sekundách a vodorovná osa počet dodavatelů. Architektura MAS je tedy v souladu s očekáváním autora mnohem efektivnější než centralizovaná architektura.



Obr. 7: Závislost doby odezvy na hloubce dodavatelského řetězce. Zdroj: (Hsieh 2019).



Obr. 8: Závislost doby odezvy na poptávce. Zdroj: (Hsieh 2019).

Pro hodnocení efektivity výrobního procesu lze použít výpočet OEE (Overall Equipment Effectiveness), tuto metodu ve své práci popisují autorky (Plinere a Aleksejeva 2019) a poprvé ji použil Seiichi Nakajima v 60. letech dvacátého století. Podle autorek OEE rozděluje výkon výrobní jednotky na tři samostatné měřitelné části:

1) Dostupnost.
2) Výkon (Performance) je procentuální vyjádření počtu skutečně vyrobených produktů vůči počtu vyrobených výrobků stanovených normou.
3) Kvalita (Quality) je procentuální vyjádření počtu výrobků, které úspěšně prošly kontrolou kvality vůči celkovému počtu výrobků.

$$e = d \cdot v \cdot k \qquad (2)$$

e OEE skóre

d Dostupnost

v Výkon

k Kvalita

Pro výpočet OEE se používá vzorec (2). OEE skóre 100 % představuje podle (Plinere a Aleksejeva 2019) dokonalou výrobu, tedy: žádné odstávky, maximální rychlost, žádné zmetky. OEE skóre 85 % je podle autorek považováno za optimální a pro mnohé firmy je dosažení tohoto skóre dlouhodobý cíl. Skóre 60 % je podle autorek typické, ale naznačuje značný prostor pro vylepšení. Skóre 40 % podle autorek není neobvyklé pro firmy, které teprve začínají pracovat na sledování a zlepšování efektivity výrobního procesu.

V práci (Plinere a Aleksejeva 2019) je uvedeno několik postupů, které mohou vést ke zlepšení OEE skóre:

- Zavedení tzv. hubené výroby. Metoda je založená na eliminaci ztrát, tj. čehokoli, co nepřidá hodnotu výrobku. Ztráty lze nalézt v dopravě, inventáři, čekání, nadprodukci a ve vadách.
- Efektivní správa zásob. Minimalizace nadprodukce.
- Snížení výrobních nákladů. Výrobní náklady zahrnují náklady na dodávky surovin, plat zaměstnanců, odpisy vybavení, marketingové náklady, elektřinu atd. Zde lze snížení výrobních nákladů dosáhnout snížením doby výroby a množství závad.
- Zkrácená výrobní doba. Propracovaný harmonogram výroby může zkrátit dobu potřebnou pro výrobu.
- Efektivní využití zařízení. Znamená odstranění zpoždění a ztrát pomocí rozpracovaného plánování výroby.

2.9 Prototypování a individuální výroba

Práce (Cupek et al. 2016) se zabývá implementací MAS v oddělení prototypování v továrně Continental v Ingolstadtu. Autoři uvádějí, že v případě

malosériové výroby se často mění výrobní technologie a výrobní zařízení musí být přizpůsobena konkrétním výrobkům. Tyto změny musí podle autorů následovat organizace postupů, aby se zabránilo ztrátám způsobeným neproduktivními časovými prodlevami, nebo aby byly alespoň sníženy. Kritickými aspekty inovačních procesů se podle autorů stala doba uvedení produktu na trh a doba vývoje produktu. Jak autoři dále uvádějí, výhody plynoucí z výrobních prováděcích systémů („Manufacturing Execution Systems“, zkráceně MES) podporujících výrobní systémy mohou být pro podniky mnohem důležitější než v případě hromadné výroby. MES jsou podle autorů rozhraní zaměřená na služby, která propojují svět obchodních operací se světem výroby. Klasické MES jsou podle autorů definovány statickou hierarchií služeb a datových struktur, takže jsou velmi obtížně modifikovatelné. Jak autoři dále uvádějí, MES by měly následovat změny ve výrobě, neměly by tedy být uzavřeným a neměnným softwarem, ale musí být vytvořeny jako flexibilní a otevřené sady služeb, které interagují s fyzickým výrobním systémem. Autoři navrhují zlepšit adaptabilitu MES změnou jejich softwarové architektury z hierarchické na heterarchickou, založenou na holonech a agentech, protože model založený na holonu je flexibilnější a škálovatelnější než centrální model výroby založený na statickém uspořádání zařízení. V takovém případě musí být podle autorů centrální systém pro podporu rozhodování nahrazen lokální podporou. Jak autoři dále uvádějí, tato architektura umožňuje uživatelům systému profitovat z distribuovaného rozvrhu činností prostřednictvím simulace provádění výrobního plánu.

Výroba individuálních produktů vyžaduje podle (Klein et al. 2018) individuální výrobu. Implementace výroby unikátních produktů s použitím tradičních přístupů je podle autorů velmi složitá. Decentralizované výrobní systémy na bázi kyberfyzikálních systémů („Cyber Physical Systems“, zkráceně CPS) nabízejí podle (Klein et al. 2018; Cupek et al. 2016) realizační schéma, což by mělo snížit složitost celého systému. Jak autoři dále uvádějí, kyberfyzikální systémy mají složitější komunikaci, větší nároky na čas a koordinaci, ale z pohledu modularity jsou méně složité a přinášejí nové možnosti diagnostiky a údržby. Digitalizace výroby a nové koncepce Průmyslu 4.0 přinášejí přímou komunikaci mezi výrobními stanicemi, čímž umožňují vzájemnou výměnu informací a společné rozhodování. Jak

autoři dále uvádějí, hlavní technologický základ digitalizace tvoří CPS. CPS podle autorů sestávají z fyzického systému (mechanická a elektrická část) a z virtuální reprezentace systému ve formě softwarových komponent, např. řídicích jednotek. Jak autoři dále uvádějí, sítě CPS jsou tvořeny množstvím komunikujících mikropočítačů. V takzvané inteligentní továrně je podle autorů zavedena velmi vysoká úroveň automatizačních systémů, která využívá flexibilní sítě výrobních kyberfyzikálních systémů („Cyber Physical Production System“, zkráceně CPPS). Jak autoři dále uvádějí, CPPS jsou schopné řídit výrobní proces do značné míry autonomně. Podle (Klein et al. 2018; Monostori 2014) CPS mohou využívat strojové učení a umělou inteligenci.

2.10 Doporučení pro úspěšnou replikaci MAS

Práce (Kamiński a Szufel 2013) upozorňuje na možnost vzniku chyb při pokusu o zopakování implementace některé z prací, které se agentovým modelováním zabývají. Autoři dále uvádějí, že pokud při pokusu o provedení stejné simulace jiným týmem vědců nebyl k dispozici přesný algoritmus použitý v původní práci, tak bylo nutné postupovat pouze podle verbálního popisu. To však podle autorů často vedlo k výsledkům, které se podstatně lišily od původní práce Východiskem z této situace je podle (Kamiński a Szufel 2013; Ormerod a Rosewell 2009) provedení dvoufázové kontroly, první fáze je nazývána verifikace a druhá validace.

Verifikace je podle (Kamiński a Szufel 2013; Ormerod a Rosewell 2009) kontrola výskytu chyb implementace simulačního modelu, tedy zda zdrojový kód neobsahuje nějaké chyby. (Ormerod a Rosewell 2009) tuto definici dále rozšiřují o *„proces určení, zda jsou rovnice správně vyřešeny“*. Pro nalezení chyb v implementaci lze podle (Kamiński a Szufel 2013) použít např. jednotkové testy, nebo nástroje vývojového prostředí IDE.

Validace podle (Kamiński a Szufel 2013; Ormerod a Rosewell 2009) spočívá ve vyhodnocení, zda simulovaný model odpovídá skutečnému prostředí. Tato definice je v práci (Ormerod a Rosewell 2009) dále rozšířena o tvrzení, že validace je *„proces určení, že používáme správné rovnice“*.

Pro úspěšnou replikaci MAS (Kamiński a Szufel 2013) doporučují v dokumentaci typického modelu založeného na agentech použít pouze následující tři typy diagramů UML (Unified Modelling Language):

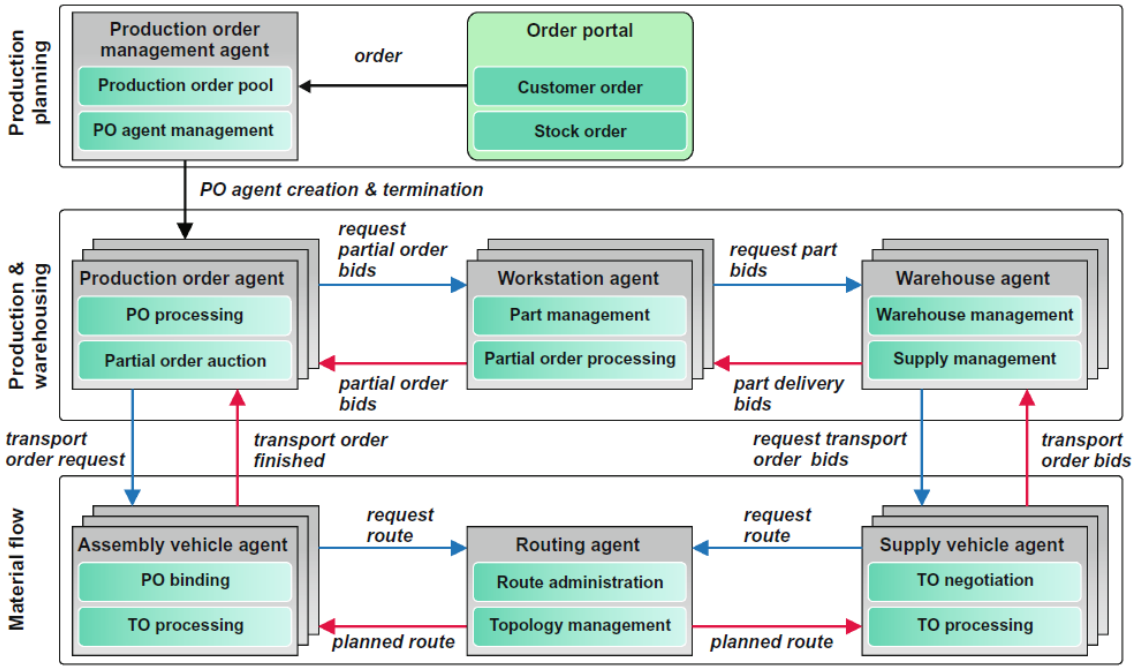
1) Diagramy tříd pro strukturu modelů na vysoké úrovni, zachycující všechny typy agentů s jejich akcemi a atributy.
2) Diagramy aktivity pro celkovou dynamiku modelu.
3) Stavové diagramy vysvětlující podrobnou dynamiku atributů pro jednotlivé třídy agentů.

V práci (Kamiński a Szufel 2013) je dále doporučeno použít v dokumentaci pseudokód, protože na základě diagramu tříd jsou sice známy metody, ale není striktně definována jejich implementace, pak může i malá odlišnost v implementaci vést k chybným závěrům. Ani to však podle autorů nemusí vždy stačit, proto je někdy nutné k práci připojit zdrojové kódy. Autoři dále zdůrazňují, že by měly být v průběhu ověřování využity veškeré možnosti testování, tedy jednotkové testy i nástroje vývojových prostředí. Kromě toho autoři doporučují řádně okomentovat a zdokumentovat veškeré zdrojové kódy, tedy vlastní implementaci modelu, jednotkové testy, kód simulačního experimentu i kód datové analýzy.

3 Metodický postup řešení

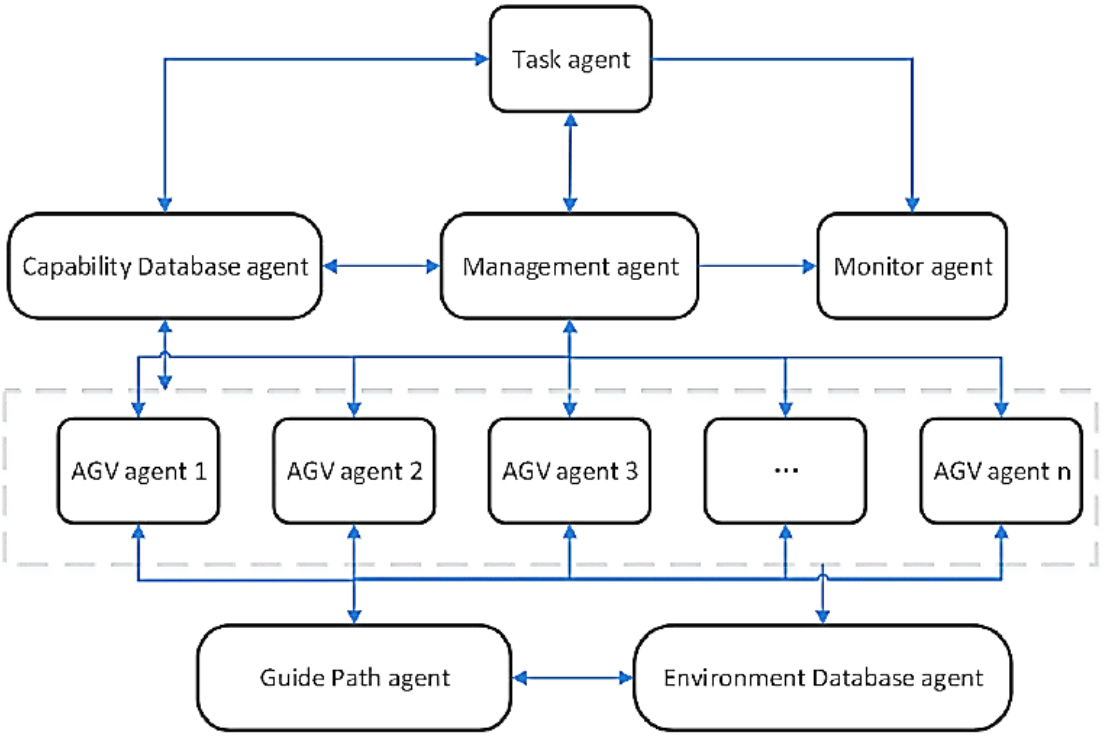
3.1 Druhy agentů ve výrobně logistických systémech

Vývoj MAS může být podle (Blesing et al. 2017) rozdělen do tří úrovní hierarchie, viz Obr. 9. Nejvyšší vrstva se podle autorů nazývá plánování výroby („*Production planning*“) a zabývá se správou objednávek a podřízenými plánovacími procesy. Tato vrstva podle autorů dále obsahuje rozhodovací algoritmus, který je součástí agentu řízení výroby („*Production Order Management agent*“, zkráceně POM-Agent) a slouží pro rozhodování, která objednávka z objednávkového portálu se načte do systému jako další. Druhá vrstva se podle autorů nazývá výroba a skladování („*Production and Warehousing*“). Tato vrstva podle autorů obsahuje agenty zabývající se fyzickým výrobním procesem v kombinaci s dodávkou komponent a řízením zásobování. Jak autoři dále uvádějí, tato vrstva obsahuje další dva rozhodovací algoritmy. Jeden je podle autorů umístěn v agentu výrobní objednávky („*Production Order agent*“, zkráceně PO-Agent) a zabývá se rozhodováním, která dílčí objednávka z grafu priorit bude vytvořena jako další. Druhý rozhodovací algoritmus se podle autorů nachází ve skladovém agentu („*Warehouse agent*“) a odpovídá za přiřazení objednávky přepravy. Nejnižší vrstva v hierarchii se podle autorů nazývá tok materiálu („*Material flow*“). Tato vrstva je podle autorů zodpovědná za plánování, správu a provádění toku materiálu mezi skladem a pracovními stanicemi. Autoři dále uvádějí, že hlavním prvkem této vrstvy je směrovací agent („*Routing agent*“).



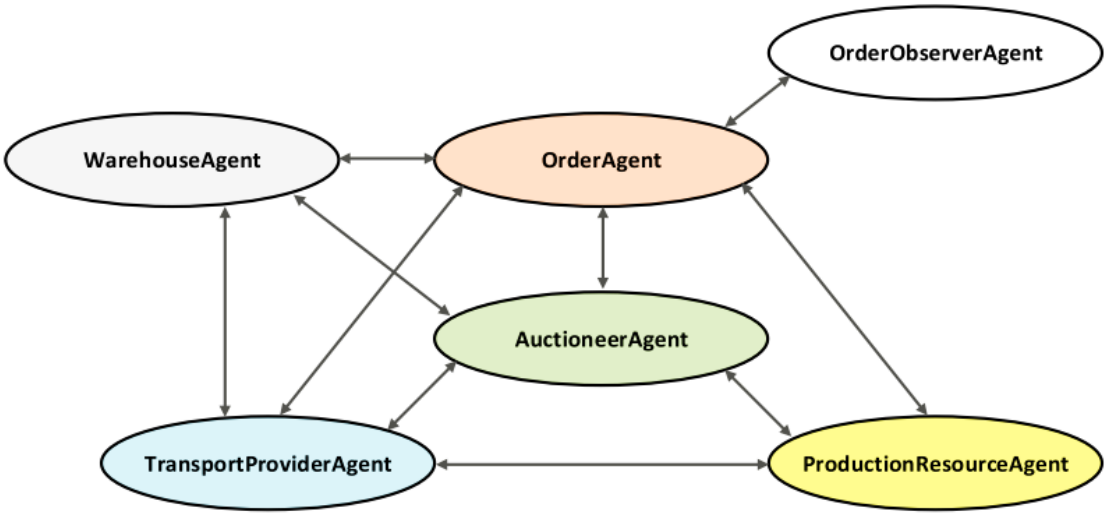
Obr. 9: Schéma decentralizovaného výrobního MAS pro automobilový průmysl. Zdroj: (Blesing et al. 2017).

V práci (Lu et al. 2019) je analýza funkční struktury systému, podobně jako v práci (Blesing et al. 2017), rozdělena do tří vrstev. V práci (Lu et al. 2019) je však rozdělení do vrstev autory koncipováno jinak, v tomto případě se jedná o vrstvu řídícího centra, vrstvu AGV s inteligentním rozhodováním a vrstvu prostředí. Vrstva prostředí podle autorů obsahuje lokalizační informace, přepravní informace atd. Jak autoři dále uvádějí, konstrukce agentu by měla splňovat požadavek na úplnost funkcí systému PL a jednotlivé funkce by měly být správně rozděleny, aby vyhovovaly potřebám snadného ovládání. Autoři navrhují hybridní multiagentovou řídicí strukturu, viz Obr. 10, která zaručuje flexibilitu a přizpůsobivost a zároveň zjednodušuje strukturu systému a snižuje složitost provozu a řízení systému.



Obr. 10: Schéma výrobně logistického MAS. Zdroj:(Lu et al. 2019).

V práci (Klein et al. 2018) je použito plánování výroby založené na vyjednávání. Základními myšlenkami jsou podle autorů inteligentní produkty, konkurence výrobních zdrojů a rychlé automatické plánování krok za krokem. Jak autoři dále uvádějí, definice požadovaných vlastností výrobku, včetně výrobního postupu, je od zákazníka přijata ve strojově čitelném formátu. Autoři ve své práci usilují o pokročilé vyjednávání mezi inteligentními výrobními zakázkami a výrobními zdroji. Celková struktura MAS navrženého autory za tímto účelem je znázorněna na Obr. 11, agenty jsou znázorněné ovály a komunikace mezi agenty je znázorněna šipkami.



Obr. 11: Architektura MAS založeného na vyjednávání. Zdroj: (Klein et al. 2018).

Model prezentovaný v práci (Blesing et al. 2017) rozděluje agenty podle reprezentace na agenty fyzické a logické. Dále mohou být podle autorů agenty rozděleny do dvou tříd. Agent dopravník („*Supply Vehicle Agent*“), skladový agent („*Warehouse Agent*“) a POM-Agent jsou podle autorů realizovány jako jednoduchý reflexní agent, ostatní agenty v této práci jsou realizovány jako reflexní agenty s vnitřní reprezentací modelu světa, viz (Russell a Norvig 2016).

3.1.1 Logické agenty

Logické agenty představují podle (Blesing et al. 2017) softwarovou komponentu, která reaguje na okolní vlivy. Autoři dále uvádějí, že na základě vnímaných vlivů jedná agent v rámci svého chování autonomně a snaží se dosáhnout svých předdefinovaných cílů. Dále je uveden stručný popis činností jednotlivých logických agentů použitých v práci (Blesing et al. 2017) a srovnání s agenty použitými v pracích (Lu et al. 2019; Klein et al. 2018):

- Production Order Management agent: POM-Agent je podle (Blesing et al. 2017) schopen přistupovat k sadě konkrétních zakázek v objednávkovém systému. Jedním z klíčových úkolů tohoto agentu je podle autorů určit, která zakázka má být spuštěna jako další. Aby bylo možné učinit toto rozhodnutí v rámci rozhodovacího algoritmu, je podle autorů agent vždy informován o

aktuálním využití celého systému včetně míry vytížení agentů pracovních stanic. Jak autoři dále uvádějí, dalším úkolem tohoto agentu je spouštět konkrétní PO-Agenty, které reprezentují vybrané zakázky a ukončit je po dokončení montáže.

- Task agent: V práci (Lu et al. 2019) podobnou funkci zajišťuje úkolový agent. Ten je podle autorů odpovědný za přebírání přepravních úkolů ze skladu, nebo od výrobního zařízení a za rozklad a integraci úkolů podle jejich velikosti, typu, doby provozu, počátečního a koncového bodu každého úkolu atd. Kromě toho úkolový agent podle autorů může hodnotit požadavky úkolu a ukládat data do agentu databáze schopností spolu s informacemi o skladu nebo zařízení, které úkol vygenerovalo, aby bylo možné rychle vybrat odpovídající AGV a snížit redundantní komunikaci.
- Order agent: Největší význam má podle autorů v práci (Klein et al. 2018) objednávkový agent. Tento agent představuje podle autorů inteligentní výrobní zakázku, která vyjednává svou cestu výrobním systémem s různými agenty produkčních zdrojů a agenty poskytovateli dopravy. Jak autoři dále uvádějí, každý objednávkový agent je pozorován agentem pozorovatelem objednávky.
- Production Order agent: Prvním úkolem PO-Agentu po spuštění zakázky je podle (Blesing et al. 2017) zahájení aukce za účelem nalezení vhodného montážního dopravníku. Jak autoři dále uvádějí, nabídky od všech dostupných agentů montážních dopravníků obsahují např. následující informace: stav baterie, vzdálenost a doba jízdy do místa montáže karoserie. Na základě těchto informací si podle autorů může PO-Agent vybrat nejlepší montážní dopravník pro dosažení globálního cíle (tím je zkrácení jízdních vzdáleností, nebo doby transportu). Hlavním úkolem PO-Agentu je podle autorů dokončit všechny dílčí zakázky daného grafu priorit. Jak autoři dále uvádějí, pro všechny otevřené a proveditelné dílčí zakázky budou následně draženy pracovní stanice. V rámci procesu hodnocení bude podle autorů vyhodnocena nejlepší nabídka pro dílčí zakázku, aby bylo dosaženo globálního cíle (tím je např. maximální využití speciálních pracovních stanic,

jednotné využití všech pracovních stanic, nebo nejkratší doba výroby zakázek).

- Management agent: V práci (Lu et al. 2019) obdobnou funkci zastává řídící agent ve spolupráci s agentem databáze schopností. Podle autorů řídící agent přijímá úkoly od úkolového agentu a je odpovědný za přiřazení transportních úkolů jednotlivým AGV agentům, které mají schopnost úkol dokončit. Řídící agent se podle autorů rozhoduje na základě informací získaných od agentu databáze schopností. Jak autoři dále uvádějí, tento agent rovněž zajišťuje předání informací o dokončení úkolu AGV agentem monitorovacímu agentu.
- Auctioneer agent: Aukční agenty v práci (Klein et al. 2018) zprostředkovávají zakázky pro agenty poskytovatele dopravy a agenty produkčních zdrojů. Tyto agenty zahajují aukce a shromažďují nabídky, jsou tedy obdobou PO-agentů.
- Capability Database agent: Agent databáze schopností podle (Lu et al. 2019) zajišťuje uložení hodnocení každého skladovacího a výrobního zařízení a jejich vliv na zpracování úkolů. Jak autoři dále uvádějí, tento agent obsahuje i schopnost dokončit přepravní úkoly AGV agentů. Vstupní data pro agent databáze schopností jsou podle autorů zadávána a aktualizována prostřednictvím úkolového agentu a AGV agentů.
- Warehouse agent: Povinnosti skladového agentu zahrnují podle (Blesing et al. 2017) správu dílčích dodávek, přiřazení zásobovacích dopravníků a správu zásob. Pro účely správy zásob má podle autorů skladový agent implementované rozhraní pro externí skladový systém. Jak autoři dále uvádějí, přepravní zakázky jsou draženy u zásobovacích dopravníků. Na základě nabídek lze podle autorů vybrat optimální zásobovací dopravník (nejkratší trasa, nejkratší čas příjezdu atd.). Podle (Klein et al. 2018) jsou skladové agenty podobné agentům produkčních zdrojů. Pokud mají produkt skladem, pak jej podle autorů nabízejí přímo.
- Routing agent: Hlavním úkolem trasového agentu je podle (Blesing et al. 2017) výpočet tras, které jsou požadovány zásobovacími a montážními dopravníky. Příchozí požadavky na přidělení trasy jsou podle autorů zpracovány sériově. Tento agent podle autorů nabízí službu, která je založena

na centralizovaném směrovacím algoritmu. Jak autoři dále uvádějí, kromě výpočtu trasy zajišťuje trasový agent i správu trasy. Výpočet tras podle autorů probíhá na základě znalosti topologie a uložených předchozích naplánovaných tras.

- Guide Path agent: V práci (Lu et al. 2019) obdobnou funkci zajišťuje agent průvodce ve spolupráci s agentem databáze prostředí. Úkolem agentu průvodce je podle autorů najít nejkratší cestu a zabránit kolizím, za tím účelem komunikuje s AGV agenty. Autoři dále uvádějí, že tento agent prostřednictvím uvažování a rozhodování hledá a vyhodnocuje nejkratší cestu, jejímž výběrem může zabránit kolizím. Agent podle autorů vyhodnocuje všechny možné cesty, kterými lze AGV vést. Jak autoři dále uvádějí, počet AGV v přepravním systému je velký a jejich umístění se mění v reálném čase, proto je tento proces vyjednávání aktualizován, jakmile AGV projede každou křižovatkou.
- Environment Database agent: Informace o prostředí v práci (Lu et al. 2019) shromažďuje a udržuje agent databáze prostředí. Podle autorů se jedná především o stálé základní informace o výrobním závodu, včetně umístění všech skladů a výrobních zařízení a informací o trasách mezi těmito lokalitami. Kromě základních informací tento agent podle autorů udržuje i aktuální informace o poloze všech AGV, které jsou získávány prostřednictvím komunikace s AGV, díky tomu je obsazenost trasy aktualizována v reálném čase.
- Monitor agent: Dohled nad přepravním systémem je podle (Lu et al. 2019) zajištěn monitorovacím agentem. Tento agent získává podle autorů informace o spuštění úlohy od úkolového agentu a zpětnou vazbu o dokončení úlohy od řídícího agentu. Jak autoři dále uvádějí, monitorovací agent umožňuje vizualizovat výsledky testů, aby správce systému rozuměl stavu systému, mohl najít případné problémy a provést včasné přizpůsobení.
- Order Observer agent: Agent pozorovatel objednávky přebírá v práci (Klein et al. 2018) roli zákazníka, který kontroluje stav všech svých objednávek, proto chce znát stav všech svých objednávkových agentů. Tento agent podle autorů také shromažďuje zprávy o výsledcích provedených výrobních kroků,

které se používají k rozpoznání spolehlivosti výrobních zdrojů. Za tímto účelem vytváří podle autorů agent pozorovatel objednávky hodnocení každého známého vyjednávacího partnera. Tato hodnocení pak mohou být podle autorů použita k tomu, aby obchodním partnerům a spolehlivým výrobním zdrojům poskytla malý bonus při hodnocení jejich nabídek.

3.1.2 Fyzické agenty

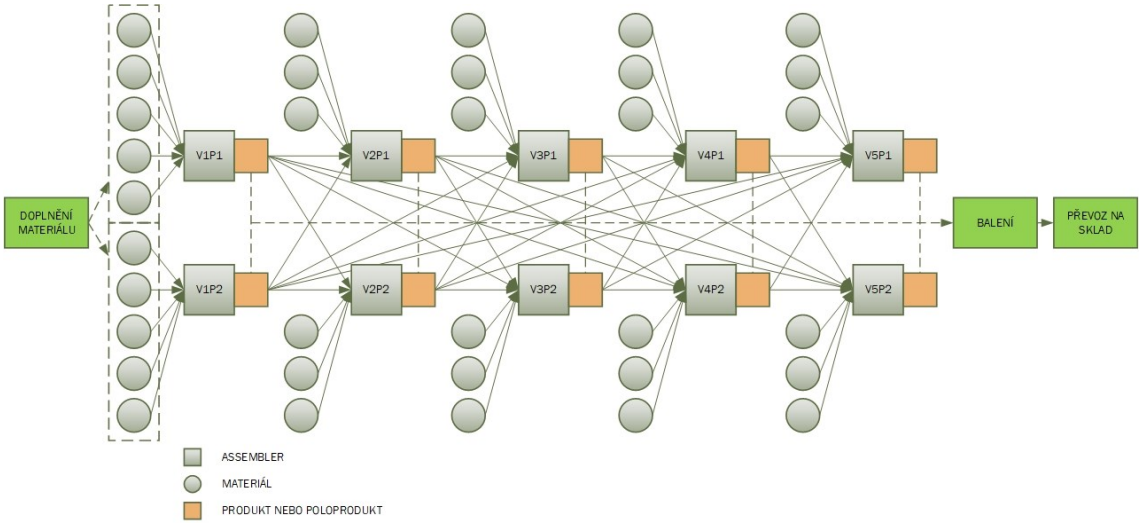
Fyzické agenty rozšiřují podle (Blesing et al. 2017) logické agenty o technické součásti, nebo je mohou propojit s externí řídící jednotkou. Fyzické agenty mohou být podle autorů spuštěny přímo na konkrétním hardware (např. řídící jednotka dopravníku), nebo na vyhrazeném PC. Dále je uveden stručný popis činností jednotlivých fyzických agentů použitých v práci (Blesing et al. 2017) a srovnání s agenty použitými v pracích (Lu et al. 2019; Klein et al. 2018):

- Vehicle agent: Agent dopravník je fyzický prostředek, který je přímo spojený s robotem, v tomto případě s dopravníkem (v originále *„Cellular Transport Vehicle“*, zkráceně CTV), více informací o CTV lze nalézt v práci (Kamagaew et al. 2011). CTV je schopen měnit své umístění, nést zásobníky, poskytovat informace o stavu (např. informace o poloze a baterii) a přijímat externí přepravní objednávky. Agent dopravník také umožňuje převést vypočtené trasy od trasového agentu na příkazy pro řízení, kromě toho může přijímat cíle tras od jiných agentů a poskytovat informace o dopravníku dalším agentům. V práci (Blesing et al. 2017) existují dvě varianty základního dopravního agentu, přičemž každý odvozený agent doplňuje další schopnosti, aby mohl plnit úlohy zásobovacího nebo montážního dopravníku.
- Agent AGV: Jádrem výrobně logistického systému a obdobou agentu dopravníku je podle (Lu et al. 2019) agent AGV, který získává transportní úlohy interakcí s řídícím agentem a informace o výběru cesty interakcí s agentem průvodcem.

- Transport Provider agent: V práci (Klein et al. 2018) zajišťují logistiku v produkčním systému agenty poskytovatelé dopravy a jsou tedy obdobou agentů dopravníků. Tyto agenty jsou podle autorů používány poskytovateli dopravy pro nabídku řešení dodávek. Jak autoři dále uvádějí, všechny agenty poskytovatelé dopravy spolu soutěží.
- Workstation agent: Agent pracovní stanice může být reprezentován dvěma různými fyzickými typy pracovních stanic. Jednou z nich je robot, který provádí montáž součástí. Druhou je pracovní stanice, kde montáž provádí zaměstnanec továrny. Pracovníci spolupracují se systémem prostřednictvím mobilního zařízení (např. tablet), kde jsou zobrazeny otevřené úkoly a zároveň je poskytnuta možnost interakce. Důležitým úkolem agentu pracovní stanice je spravovat vlastní lokální stav zásob. Pokud úroveň zásob klesne pod definovanou úroveň, automaticky se spustí opětovné objednání. Pro provedení dílčích objednávek musí být agent pracovní stanice schopen vytvořit nabídky pro tyto objednávky, které jsou následně draženy PO-Agentem. V případě, že požadovaná součást není k dispozici lokálně, bude objednána prostřednictvím skladového agentu. Všechny potřebné informace, jako je dostupnost nebo doba přepravy, jsou shromažďovány a poskytovány konkrétnímu agentu pracovní stanice.
- Production Resource agent: Agent produkčního zdroje je v práci (Klein et al. 2018) obdobou agentu pracovní stanice. Podle autorů je každý zúčastněný produkční zdroj reprezentován tímto agentem. Jak autoři dále uvádějí, agent se může účastnit aukcí i přímých jednání a snaží se získat co nejvíce výrobních zakázek, aby maximalizoval svůj zisk. Výsledkem je podle autorů to, že všechny agenty produkčních zdrojů spolu soutěží.

3.2 Popis experimentu a cíl práce

Experiment je zaměřen na vyhodnocení výpadků dodávek materiálu, nebo polotovarů ve víceinstanční průmyslové výrobě. Pro potřeby experimentu bylo nutné vytvořit model dodavatelského řetězce skládajícího se z deseti výrobních linek, viz Obr. 12. Možné vazby mezi jednotlivými výrobními linkami jsou pro lepší přehlednost v obrázku pouze naznačeny, ve skutečnosti může model obsahovat i další vazby, viz níže.



Obr. 12: Model výrobního řetězce. Zdroj: výsledek konzultací autora s vedoucím práce.

Model výrobního procesu obsahuje dvě vzájemně provázané pětistupňové sekvence bez laterálních vazeb. Každý stupeň v obou sekvencích umožňuje zapínání a vypínání výstupů jednotlivých instancí za účelem simulace výpadků dodávek. Model lze formálně popsat pomocí následující matice:

$$\begin{bmatrix} V_1P_1 & V_2P_1 & V_3P_1 & V_4P_1 & V_5P_1 \\ V_1P_2 & V_2P_2 & V_3P_2 & V_4P_2 & V_5P_2 \end{bmatrix}$$

V_x vrstva výroby

P_y produkční linka

Libovolný polotovar vzniklý ve vrstvě V_x může být použit v kterékoliv z následujících vrstev $V_{(x+1)}$ až V_5. Každý proces skrytý v uzlu V_xP_y má přiřazenou

určitou časovou náročnost zpracování $t > 0$, která je navýšena o čas potřebný pro přepravu materiálu, polotovarů a produktů (vč. přepravy v rámci výrobní haly).

Každý produkt vyrobený na výrobních linkách ve vrstvě V_1 se skládá z pěti typů výrobních surovin dodaných externími dodavateli. Vzniklé polotovary se postupně zpracovávají v dalších vrstvách. Každá výrobní linka od vrstvy V_2 do vrstvy V_5 vytváří své produkty ze dvou až pěti surovin dodaných nižšími vrstvami a dvou až tří surovin dodaných externími dodavateli. Volba počtu vstupů jednotlivých výrobních linek, výběr dodavatele pro každý z těchto vstupů a nastavení počtu všech typů surovin nutných pro výrobu jednoho produktu bude proveden generátorem pseudonáhodných čísel. Při generování výše uvedených vlastností linek a jejich vzájemného propojení je průběžně dopočítáváno množství produktů, které musí každá linka vyrobit, aby byl dodavatelský řetězec schopný pokrýt poptávku všech zákazníků. V opačném případě by se mohlo stát, že některá výrobní linka nebude stíhat dodávat včas suroviny vyšším vrstvám, nebo bude mít naopak vysokou nadprodukci. Nerovnováha výrobního řetězce by se také negativně projevila z pohledu ekonomiky výroby. Hotové výrobky si zákazníci přebírají z výstupních skladů výrobních linek v okamžiku, kdy je pokrytá jejich objednávka.

Všechny výrobní linky v dodavatelském řetězci mají v počátečním stavu vstupní sklady surovin naplněné na úroveň stanovenou nastavením parametrů modelu. V práci (Tucnik et al. 2017) je předpokládáno, že před začátkem výroby jsou v hlavním skladu k dispozici komponenty dostačující alespoň pro jednu směnu a také je předpokládáno, že jsou komponenty pro výrobu připravené na výrobních linkách. Tato strategie je výhodná, protože zkracuje čas potřebný pro dosažení optimálního vytížení všech výrobních linek. Tím je zároveň zkrácená i doba běhu simulace výroby, neboť vyšší vrstvy výrobních linek nemusí čekat až nižší vrstvy vyprodukují dostatek surovin nezbytných pro zahájení výroby ve vyšších vrstvách.

V současnosti se management většiny podniků snaží minimalizovat zásoby surovin a polotovarů, protože skladováním většího množství materiálu se zvyšují výrobní náklady a tím i cena finálního produktu. V modelu je možné nastavit maximální kapacity vstupních i výstupních skladů. V rámci dodavatelského řetězce se dá očekávat společná strategie, takže toto nastavení ovlivňuje všechny výrobní linky v dodavatelském řetězci.

V práci (Tucnik et al. 2017) je dále uvažováno, že zaměstnanci pracují ve směnném provozu, tato skutečnost je modelována následujícím způsobem. Za normálních okolností pracují výrobní linky s efektivitou výroby 100 %. Efektivita výroby 100 % odpovídá pěti pracovním dnům v týdnu (po – pá) v třísměnném provozu. Efektivitu výroby vybraných instancí je možné krátkodobě navýšit až na 120 %. Dvacetiprocentní navýšení efektivity výroby odpovídá zavedení pracovních sobot (také v třísměnném provozu), tedy celkem šesti pracovních dní (po – so). Kromě toho model umožní regulaci produktivity, aby bylo možné plynule regulovat rychlost výroby, tak jako tomu je v reálném provozu.

Zásobování všech vstupů všech výrobních instancí probíhá při poklesu množství surovin ve vstupní frontě pod nastavitelnou úroveň. Tato úroveň je uvažována jako rezerva určená pro zajištění pokračování výroby při případném zpoždění logistiky. Při vyčerpání této rezervy bude nutné odstavit výrobní linku, na jejímž vstupu došlo ke vzniku chyby zásobování. Vznik chyby kdekoliv v modelu dodavatelského řetězce bude mít dopad i na všechny navazující uzly. V konečném důsledku tedy může dojít i ke zpoždění dodání hotového produktu zákazníkům, protože kapacita výstupních skladů je, stejně jako kapacita skladů surovin a polotovarů, omezená na minimum. Toto omezení vyplívá z výše uvedené snahy o minimalizaci výrobních nákladů. Zpoždění dodání produktu může způsobit další náklady spojené s penalizací, případně může způsobit přesun zákazníků ke konkurenci. Proto je snahou každého podniku maximalizovat efektivitu logistiky a zabránit tak výpadkům dodávek. S tím souvisí i snaha o maximalizaci intervalů mezi plánovanými odstávkami a minimalizaci doby trvání plánovaných i neplánovaných odstávek. Po obnovení dodávek surovin je možné na základě objednávek zvýšit efektivitu výroby. Zvýšení efektivity by mělo zajistit urychlení dodání polotovarů pro navazující výrobní instance a tím minimalizovat trvání neplánovaných odstávek, nebo v ideálním případě odstávky úplně eliminovat.

Cílem experimentu bude vyhodnocení dopadů chyb na kontinuitu výroby a testování různých strategií, které by měly chránit výrobní řetězec proti výpadkům. Výstupem budou tabulky a grafy zachycující šíření chyby v modelu. Na základě výstupů experimentu bude možné porovnat závislost množství produktů na času. Porovnání bude provedeno mezi normálním provozem bez výpadků logistiky a

nouzovým provozem způsobeným výpadkem dodávek. Všechny tabulky a grafy budou zobrazovat stav po ustálení počátečních výkyvů výroby.

Simulaci bude možné pro porovnání výsledků spustit v režimu bez poruchy, nebo s poruchou. Režim s poruchou může být spuštěn bez opravy této poruchy, nebo s opravou. Porucha bude reprezentovat libovolný typ chyby (např. logistická chyba, závada stroje atd.). V rámci experimentu budou testovány různé strategie, které budou chránit model proti výpadkům. Simulace chyby způsobí propad a opatření jej má napravit. Nepůjde tedy o preventivní opatření, ale spíše o nápravu vzniklé situace. Nápravu lze řešit navýšením zásob ve skladu, zavedením dodatečných směn v případě výpadku, dovezením polotovaru z jiné pobočky, aby se nahradil výpadek apod.

4 Praktická část

Podle (Tanajura et al. 2015) byly multiagentové systémy úspěšně použity pro reprezentaci distribuovaných výrobních systémů. Autor dále uvádí, že agenty pomáhají snížit množství informací určených operátorovi a tím mu umožní věnovat více času k tomu, aby se zaměřil na situace, které vyžadují náležitou pozornost. Což má za následek zvýšení efektivity.

4.1 Software použitý pro vytvoření modelu

AnyLogic je multiplatformní software s propracovaným GUI umožňující simulační modelování. K modelování používá metodiky založené na agentech, diskrétních událostech a dynamice systému. Tato práce je zaměřená na multiagentové systémy, proto bylo možné využít k vytvoření experimentálního modelu AnyLogic. Tento software využívá programovací jazyk Java, a to nejen pro vlastní běh prostředí, ale i pro vytvořený model. Díky tomu je možné doplněním Java kódu ovlivnit chování modelu a přizpůsobit jej vlastním potřebám. Pro uložení proměnných veličin lze v AnyLogic použít parametry a proměnné. Pro větší přehlednost jsou parametry v modelu použité pouze pro prvotní nastavení modelu. Všechny hodnoty, které se v průběhu simulace budou měnit jsou uložené v proměnných.

V této práci byla použita verze Personal Learning Edition (dále jen PLE), která má určitá omezení. Například není možné v modelu vytvořit více než 10 typů agentů. S ohledem na možnosti budoucího rozšíření modelu tedy muselo dojít k určitým kompromisům, které jsou v rozporu s doporučeným způsobem návrhu objektového modelu. Pro vytvoření výrobních linek a externích dodavatelů by bylo vhodné využít dědičnost, tím by však došlo k navýšení použitých typů agentů. Dalším omezením verze PLE je omezení maximálního počtu agentů v systému, přičemž mezi agenty je počítána i každá surovina a každý produkt vytvořený výrobní linkou. Proto jsou výrobní linky navržené s využitím programovacího jazyka Java a není při jejich návrhu použito grafických prvků modelu. Mezi výhody AnyLogic patří možnost vytvoření vlastních tříd. Prvky modelu, které nutně

nevyžadují chování, nebo vlastnosti agentu (např. autonomii) tedy mohou být modelovány jako třída. Tím jsou sníženy nejen nároky na počet agentů, nebo typů agentů, ale i výpočetní náročnost simulace.

Kromě jednotlivých agentů a tříd obsahuje AnyLogic komponentu *Simulation* a vlastní databázi. *Simulation* slouží k počátečnímu nastavení všech vlastností modelu před spuštěním vlastní simulace. Globální vlastnosti ovlivňují nejen vlastní model, ale i běhové prostředí Java, ve kterém simulace probíhá. Jedná se například o maximální velikost paměti alokované pro model. Mezi další důležité vlastnosti patří nastavení modelového času, tedy počátečního a koncového data běhu simulace a poměru rychlosti běhu simulace ku skutečnému času. Další důležitou vlastností je možnost nastavení generátoru pseudonáhodných čísel. Pro simulaci zde lze nastavit náhodné semínko, nebo fixní semínko. Náhodné semínko se využívá, pokud má být každá simulace unikátní. Fixní semínko lze nastavit v případě požadavku na reprodukovatelnost spouštění simulace. Databázi AnyLogic lze využít k uložení a načtení informací o modelu, například počátečního nastavení, seznamu agentů atd. Databáze umožňuje import dat z externích zdrojů, např. Microsoft SQL databáze, nebo souborů vytvořených v aplikaci Microsoft Excel, případně Access.

4.2 Model výrobního řetězce

Model výrobního řetězce navržený pro experimentální ověření výpadků logistiky se skládá z následujících komponent:

- Typy agentů:
  - *Main* (základní část modelu): Tvoří základ modelu a propojuje všechny ostatní komponenty.
  - *ProductionLine* (výrobní linka): Zajišťuje výrobu, objednávky a nastavení optimální produktivity.
  - *Customer* (koncový zákazník): Objednává finální produkty od jednotlivých výrobních linek.
  - *Truck* (kamion): Zajišťuje dopravu mezi výrobními linkami a od externích dodavatelů.
- Pomocné třídy:

- *Input* (vstup): Reprezentuje vstup výrobní linky.
- *ProductOrder* (objednávka produktů): Představuje objednávku produktů, nebo surovin pro další výrobu
- *TransportOrder* (objednávka dopravy): Reprezentuje objednávku dopravy.

- Ostatní komponenty:
  - *Simulation*: Umožňuje prvotní nastavení parametrů, s kterými bude následně spuštěna simulace.
  - *Database*: Obsahuje základní nastavení některých agentů.

Zdrojové kódy použité v modelu jsou v příloze č. 2–7. Kompletní model v elektronické podobě je v příloze č. 8.

4.2.1 Agent *main*

Agent *main* existuje v modelu pouze v jedné instanci odvozené od typu agentu *Main*. Tento agent obsahuje především dva pohledy, které jsou nezbytné pro běh simulace a vyhodnocení experimentu a také menu sloužící pro přepínání mezi těmito pohledy.

První pohled obsahuje GIS mapu a v menu je nazvaný *„Animace“*. Na mapě je na základě adres uložených v databázi rozmístěno deset výrobních linek (v1p1 až v5p2), dále je na mapě rozmístěno jedenáct externích dodavatelů (ext0 až ext10) a depo kde parkují kamiony. Externí dodavatelé jsou stejným typem agentů jako výrobní linky, ale neřeší se u nich nastavení produktivity, ani vlastní výroba atd. Zajišťují pouze zásobování pro vstupy výrobních linek vXpX, jejich zásoby jsou neomezené. Další funkcionality externích dodavatelů nejsou nutné pro běh experimentu, proto se od nich v modelu abstrahuje.

Druhý pohled (výchozí) je v menu modelu nazvaný *„Statistiky“* a obsahuje pět grafů. Každý graf reprezentuje jednu vrstvu modelu, tedy charakteristiky dvou výrobních linek. Grafy zobrazují závislosti množství výrobků ve výstupních frontách na čase. Z grafů je možné exportovat tabulky s hodnotami jednotlivých veličin.

Dále agent *main* obsahuje čtyři kolekce agentů nazvané *externalSuppliers*, *productionLines*, *trucks* a *customers* a kolekci objednávek dopravy nazvanou *transportOrders*. Tyto kolekce budou popsané v dalších kapitolách.

Funkce *sendTrucks* zajišťuje přidělení úkolů agentům typu *Truck*. Dokud existuje alespoň jeden požadavek na dopravu v kolekci *transportOrders* a zároveň existuje alespoň jeden volný kamion, tak funkce *sendTrucks* postupně odesílá kamiony vyřídit požadavky.

Funkce *generateSupplyChain* zajišťuje prvotní inicializaci výrobních linek. Jeden výstup linky V_xP_y zásobuje vždy jeden vstup linky $V_{(x+1)}P_y$. Počty ostatních vstupů a jejich napojení na dodavatele jsou vygenerovány pseudonáhodně. Dodavateli jsou v tomto případě nižší vrstvy výrobních linek a externí dodavatelé. Funkce *generateSupplyChain* také pro každý vstup výrobní linky pseudonáhodně generuje počet surovin nutných pro výrobu jednoho výstupního produktu. Kromě výše uvedeného tato funkce v průběhu generování vstupů ještě dopočítává počty produktů všech výrobních linek v 1. až 4. vrstvě modelu nutných pro výrobu jednoho produktu na každé z výrobních linek v 5. vrstvě modelu a ukládá výsledek do parametru *baseProductivity* jednotlivých výrobních linek. Tato operace je nutná, protože výrobní linky V_1P_1 – V_5P_2 patří do jednoho výrobního řetězce, a musí tedy být nastavené tak, aby byl celý výrobní řetězec v optimální rovnováze. Výrobní linky v 5. vrstvě mají parametr *baseProductivity* nastavený na hodnotu 1. Funkce *generateSupplyChain* dále nastavuje parametr *overProduction*. Tento parametr ovlivňuje nadprodukci výrobních linek určenou k odběru koncovými zákazníky, proto je u výrobních linek v 1. až 4. vrstvě nastaven na hodnotu převzatou z komponenty *Simulation* a u linek v 5. vrstvě na hodnotu 1.

Dále agent *main* obsahuje parametry společné pro celý model, které se přebírají z komponenty *Simulation* v okamžiku spuštění simulace. Zde je důležitý především parametr *numberOfTrucks*, který určuje počet kamionů v modelu. Z ostatních parametrů agentu *main* se generují parametry jednotlivých agentů výrobních linek v rámci jejich inicializace, viz kapitola 4.2.2.

Funkce *initFault* a události *startFault* a *stopFault* osahují kód pro zajištění generování poruchy v průběhu simulace. Nastavení typu poruchy, výrobní linky s poruchou, dne výskytu poruchy a délky poruchy jsou uložená v parametrech

faultSimulation, *faultLine*, *faultDay* a *faultDuration*. Proměnné *faultInputNum* a *faultInputPL* slouží k dočasnému uložení informací o vstupu, který je v poruše.

Proměnná *selectedViewArea*, funkce *navigate* a grafické prvky hlavního menu obsahují pouze pomocnou funkcionalitu. Slouží pro přepínání zobrazení mezi výše zmíněnými pohledy při spuštěné simulaci.

4.2.2 Agenty *productionLine*

Jádrem celého modelu je deset instancí agentů *productionLine* odvozených od typu agentu *ProductionLine*. Názvy jednotlivých instancí výrobních linek a jejich adresy (umístění na mapě) jsou importovány z databáze. V rámci inicializace celého dodavatelského řetězce funkcí *main.generateSupplyChain* je nejprve spočítán a nastaven parametr *baseProductivity*. Tento parametr stanovuje počet výrobků konkrétní výrobní linky nutných pro výrobu jednoho produktu výrobními linkami v poslední vrstvě, viz kapitola 4.2.1.

Při inicializaci výrobní linky funkcí *initParameters* je nejprve vynásoben parametr *baseProductivity* parametrem *overProduction* a následně zaokrouhlen nahoru na celé číslo a uložen zpět do parametru *baseProductivity*. Tato operace určí finální počet produktů, které je nutné vyrobit pro uspokojení potřeb odběratelů v rámci dodavatelského řetězce (výrobních linek) i potřeb koncových zákazníků. Funkce *initParameters* poté provede výpočet ostatních parametrů výrobní linky prostým násobením parametrů převzatých z agentu *main* a parametru *baseProductivity* konkrétní linky. Jedná se o následující parametry:

- *inQueueCapacity* (kapacita vstupní fronty): Určuje maximální počet kusů jednoho typu suroviny ve vstupním skladu výrobní linky.
- *inQueueMin* (minimální počet kusů ve vstupní frontě): Pokud je stav vstupní fronty nižší než hodnota tohoto parametru, dojde k vytvoření závazné objednávky surovin u dodavatele.
- *inQueueInit* (počáteční počet kusů ve vstupní frontě): Slouží pro prvotní nastavení stavu vstupních front v okamžiku spuštění simulace. V průběhu simulace již není dále využíván.

- *outQueueCapacity* (kapacita výstupní fronty): Určuje maximální počet kusů produktů ve výstupním skladu výrobní linky.
- *productsPerHour* (počet produktů za jednu hodinu): Určuje počet produktů vyrobených výrobní linkou za jednu hodinu při standardní produktivitě.

Všechny tři hodnoty parametrů týkajících se vstupní fronty určují počty surovin v poměru kusů surovin na kusy produktů 1 : 1. Pokud je počet surovin na výrobu jednoho produktu u konkrétního vstupu výrobní linky vyšší než jedna, je nutné tuto hodnotu ještě vynásobit hodnotou *numberPerProduct* uloženou v instanci daného vstupu.

Každá výrobní linka dále obsahuje následující proměnné:

- *outQueue* (výstupní fronta): Počet produktů ve výstupní frontě výrobní linky.
- *productivity* (produktivita): Proměnná *productivity* je typu double a určuje normalizovanou produktivitu výroby. Průběžné aktualizování produktivity zajišťuje funkce *setProduction*, viz níže.
- *preOrders* (předobjednávky): Tato proměnná slouží k uložení požadavků sdílených ve výrobním řetězci. Jedná se o sdílení informací o objednávkách koncových zákazníků s výrobními linkami, kterých se dané objednávky týkají. Tyto požadavky jsou šířeny postupně ve směru od vyšších vrstev k nižším. Proměnná *preOrders* slouží pouze pro nastavení produktivity výrobní linky. Závazné objednávky dorazí k nižším vrstvám s určitým zpožděním, které je dané tím, že vyšší vrstvy mají v okamžiku přijetí objednávky od zákazníků ještě dostatek surovin.
- *reservation* (rezervace): Jedná se o počet produktů rezervovaných v okamžiku objednávky dopravy. Proměnná slouží k tomu, aby nedošlo k realizaci dalších objednávek dopravy v intervalu, kdy už je objednaná doprava určitého množství produktů, ale tyto produkty ještě nejsou naložené v kamionu.
- *fault* (porucha): Na počátku simulace je u všech výrobních linek tato proměnná nastavena na hodnotu *false*. V průběhu simulace poruchy je na vybrané výrobní lince nastavena proměnné *fault* hodnota *true*.

Kolekce *customerOrders* obsahuje objednávky od koncových zákazníků. V kolekci *chainOrders* jsou uložené objednávky od vyšších vrstev dodavatelského řetězce. Toto rozdělení je důležité s ohledem na prioritizaci *chainOrders* před *customerOrders*. Ukládají se pouze závazné objednávky. Kolekce *inputs* obsahuje informace o jednotlivých vstupech, detailní popis třídy *Input* viz níže. Agent typu *ProductionLine* obsahuje následující funkce:

- *runProduction* (vlastní výroba): Tato funkce je rozdělená na tři části. První část zajišťuje závazné objednání surovin pro všechny vstupy výrobní linky. Objednávka surovin je odeslána při poklesu surovin některé vstupní fronty pod mez stanovenou vynásobením parametru *inQueueMin* a počtu surovin na výrobu jednoho produktu, který je uložen v instanci příslušného vstupu v proměnné *numberPerProduct*. Druhá část funkce slouží k objednání dopravy. Pokud jsou zásoby ve výstupní frontě dostatečné pro pokrytí objednávky, tak je dodavatelskou výrobní linkou objednána doprava a zároveň je provedena rezervace počtu produktů nutných pro pokrytí této objednávky, viz výše uvedená proměnná *reservation*. Tento krok je možné opakovat dokud je ve výstupní frontě dostatek produktů bez rezervace. Poté je zavolána funkce *main.sendTrucks*. Třetí část funkce *runProduction* spočívá v simulování výroby. Počet produktů vyrobených v následující hodině je stanoven na základě dostupné kapacity výstupní fronty a množství surovin ve vstupních frontách. Počet produktů dále ovlivňuje stav proměnné *productivity* a parametr *productsPerHour*. Poté je navýšen počet produktů ve výstupní frontě a adekvátně sníženy počty surovin ve vstupních frontách.
- *setProduction* (nastavení výroby): Úkolem funkce je plynulá regulace produktivity výroby prostřednictvím proměnné *productivity*. Tato proměnná může nabývat následujících hodnot. Hodnota 0 znamená stav, kdy nelze vůbec vyrábět (např. vyčerpání vstupní fronty, nebo plná výstupní fronta). Hodnota 1 znamená standardní, neboli stoprocentní produktivitu, na kterou je výrobní linka navržená. Maximální krátkodobá produktivita výroby byla stanovena s ohledem na kapacitu linek a maximální výkonovou zátěž zaměstnanců na 140 % standardní produktivity. Tomu odpovídá hodnota

konstanty *MAX_PRODUCTIVITY* nastavená na hodnotu 1,4. Konstanta *STEP* určuje krok změny proměnné *productivity*, aby nedocházelo ke skokovým změnám produktivity. Poté následuje kód nastavení produktivity na základě proměnné *preOrders*, aktuálního počtu nevyřízených objednávek a dostupné kapacity výstupní fronty.

- *initParameters* (inicializace parametrů): Zajištění prvotní inicializace parametrů výrobní linky. Činnost této funkce je popsána výše.

Součástí každé výrobní linky jsou dva plánovače výroby. Plánovač *weekSch* zajišťuje spouštění funkcí *setProduction* a *runProduction* každou celou hodinu v pracovní dny. Plánovač *saturdySch* spouští obě výše uvedené funkce každou celou hodinu v sobotu. Funkce *runProduction* je spuštěna pouze pokud je splněna podmínka, že proměnná *productivity* je větší než 1. To v praxi znamená zavedení sobotních směn na konkrétní lince. Toto opatření má za cíl uvést výrobní řetězec co nejrychleji zpět do rovnovážného stavu. Oba plánovače obsahují ještě podmínku, aby ke spuštění funkce *setProduction* došlo pouze u výrobních linek zapojených ve zkoumaném dodavatelském řetězci a nikoliv u externích dodavatelů. Spuštění funkce *runProduction* je dále podmíněno stavem proměnné *fault*, viz výše.

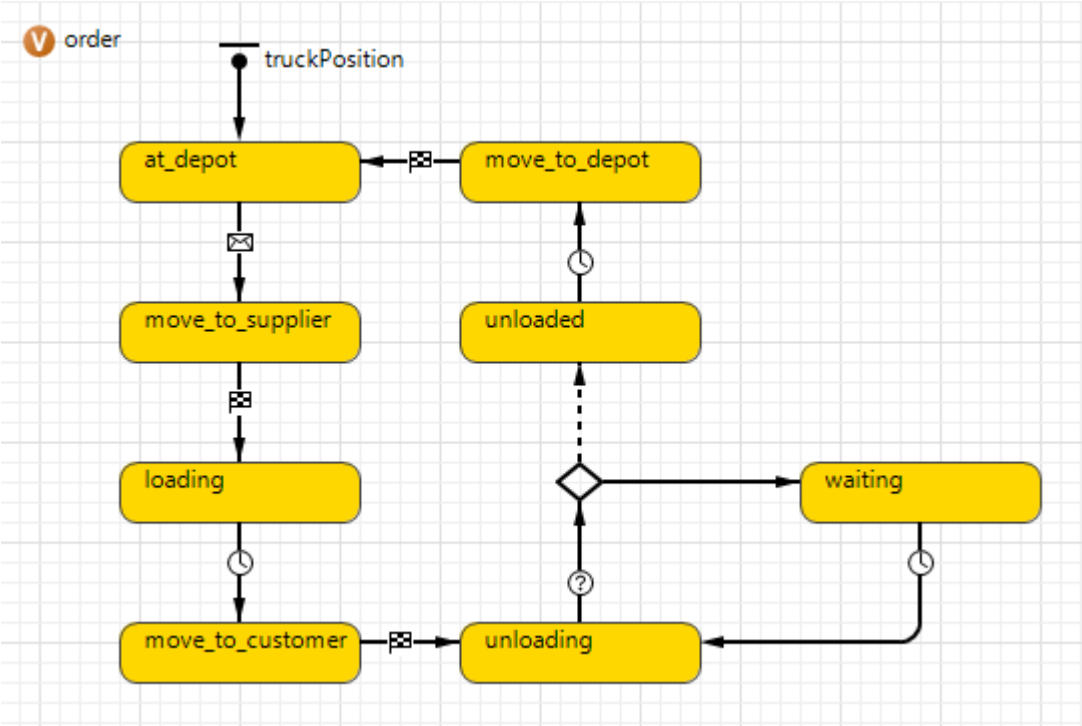
Agenty spolu komunikují prostřednictvím zasílání zpráv. Zpracování zpráv se provádí v komponentě *connections*. Agenty *productionLine* přijímají zprávy od výrobních linek ve vyšších vrstvách a od koncových zákazníků. Objednávky mohou být závazné, nebo tzv. předobjednávky. Závazné objednávky odesílají koncoví zákazníci, nebo jednotlivé výrobní linky, v případě poklesu zásob surovin pod stanovenou mez. Princip předobjednávek je popsán výše. Objednávky od koncových zákazníků a předobjednávky jsou sdíleny se všemi vstupy výrobní linky, které zásobuje některá z linek V_1P_1 – V_5P_2. Při příjmu předobjednávky je zvýšena hodnota *preOrders*. Přijatá závazná objednávka je uložena do kolekce *customerOrders*, případně *chainOrders*.

Agenty *productionLine* reprezentují nejen výrobní linky, ale i jedenáct externích dodavatelů. Externí dodavatelé mají funkcionalitu omezenou pouze na příjem objednávek produktů a zajištění objednávek dopravy (součást funkce *runProduction*).

4.2.3 Agenty *truck*

Pro zajištění přepravy produktů obsahuje model agenty *truck*, neboli kamiony. Počet těchto agentů lze nastavit před spuštěním simulace, poté již měnit nelze. Pohyb kamionů je v průběhu simulace zobrazován na mapě. Pokud dojde k vyčerpání celé flotily, musí se čekat, až se některý kamion vrátí zpět do depa.

Simulace chování kamionu je určena stavovým diagramem, viz Obr. 13. Kromě stavového diagramu obsahuje agent *truck* ještě instanci třídy *TransportOrder* nazvanou *order*. V této proměnné jsou uloženy veškeré informace týkající se vyřizované objednávky.



Obr. 13: Stavový diagram chování agentu truck. Zdroj: vlastní zpracování.

Výchozím bodem diagramu je *truckPosition*. Po spuštění simulace jsou všechny agenty *truck* umístěné v depu (stav *at_depot*), kde čekají na zprávu s objednávkou dopravy od agentů typu *ProductionLine*. Pokud agent *truck* ve stavu *at_depot* zjistí, že kolekce *main.transportOrders* není prázdná, převezme si první objednávku a spustí funkci *receive*, čímž nasimuluje příjem zprávy. Pokud je v kolekci více objednávek, pak jsou seřazeny podle času vytvoření. Kolekce reprezentuje frontu typu FiFo. Tento algoritmus zajišťuje nápravu stavu, kdy

postupně došlo k vytížení celé flotily kamionů. Každý kamion, který se vrátí do depa si díky němu automaticky přebírá jeden nevyřízený požadavek.

V reakci na příjem zprávy agent okamžitě vyráží k dodavateli. V okamžiku, příjezdu k dodavateli, je zahájeno naložení nákladu. To spočívá ve snížení počtu produktů ve výstupní frontě a snížení počtu kusů rezervovaných pro dopravu. Nakládka může trvat v rozsahu od jedné do dvou hodin a je určená funkcí *uniform*.

Po naložení nákladu se agent vydává na cestu k odběrateli, kde vyloží celý náklad, nebo alespoň jeho část. Pokud dojde k naplnění vstupní fronty na úroveň její maximální kapacity, tak přechází kamion do stavu čekání, ve kterém setrvá jednu hodinu. Poté se opět pokusí vyložit zbytek nákladu. Při vykládání je adekvátně zvýšen stav vstupní fronty a snížen stav objednávky.

Po úspěšném vyložení celého nákladu je nastavena proměnná vstupu *orderShipped* na hodnotu *false*. Tím je objednávka kompletně vyřízena a výrobní linka může v případě potřeby u tohoto vstupu zahájit generovaní další objednávky. Doba vyložení nákladu je generována funkcí uniform v rozsahu od jedné do dvou hodin. Poté dojde k přesunu do depa, kde agent převezme první nevyřízenou objednávku, nebo zahájí čekání na příjem zprávy.

4.2.4 Agent *customer*, pomocné třídy a ostatní komponenty modelu

Agent *customer* se modelu vyskytuje v deseti instancích odvozených od agentu typu *Customer*, v modelu reprezentuje koncového zákazníka. Jméno agentu je uložené v parametru *name* a sestává z textu *„cust“* následovaného indexem konkrétní instance. Dále agent obsahuje událost *generateDemand*, která je spouštěna opakovaně v rozsahu jeden až dva dny. Čas opakování je generovaný funkcí *uniform*. Tato událost obsahuje kód, který zajišťuje vygenerování objednávky a odeslání výrobní lince vybrané funkcí *uniform_discr*. Počet objednaných kusů je generován funkcí *triangular*.

Třída *Input* reprezentuje vstup výrobní linky a obsahuje následující atributy:

- *productionLine* (dodavatel): Dodavatelem může být výrobní linka zapojená v dodavatelském řetězci, nebo externí dodavatel.

- *queue* (fronta): Vyjadřuje aktuální počet surovin ve vstupní frontě.
- *numberPerProduct* (počet surovin na produkt): Obsahuje počet kusů suroviny nutných pro výrobu jednoho produktu.
- *orderShipped* (objednávka poslána): Informuje o tom, jestli již byla odeslána objednávka, aby nedošlo k nekontrolovatelnému generování objednávek.

Třída *ProductOrder* reprezentuje objednávku produktů. Tato třída je v modelu využívána k uložení informací o závazné, či nezávazné objednávce a pracují s ní především agenty typu *ProductionLine* a *Customer*. Tato třída obsahuje následující atributy:

- *customer* (zákazník): Reprezentuje odběratele a je odvozený od typu *Agent*. Tento atribut může obsahovat agent typu *ProductionLine*, nebo *Customer*.
- *input* (vstup): Pokud je odběratelem výrobní linka, pak tento atribut obsahuje vstup, pro který je určena tato objednávka. Pokud se jedná o nezávaznou objednávku, nebo je odběratelem koncový zákazník, tak je atribut *input* nevyužitý (obsahuje *null*).
- *quantity* (množství): Definuje objednané množství produktů.

Třída *TransportOrder* reprezentuje objednávku dopravy a pracují s ní agenty *productionLine*, *truck* a *main*. Tato třída je velmi podobná třídě *ProductOrder*, liší se tím, že místo atributu *customer* obsahuje dva atributy typu *ProductionLine*. Atribut *from* reprezentuje dodavatelskou výrobní linku a atribut *to* reprezentuje odběratelskou výrobní linku. Smysl atributů *input* a *quantity* je stejný jako u třídy *ProductOrder*, viz výše.

V zobrazení komponenty *Simulation* jsou připraveny posuvné ovladače a přepínače, jejichž pomocí se nastavují parametry agentů. Jedná se například o počet kamionů, nastavení výskytu poruchy a parametrů výrobních linek, viz kapitoly 4.2.1 a 4.2.2. Ve vlastnostech simulace je nastavené fixní semínko generátoru pseudonáhodných čísel, aby byla zajištěna reprodukovatelnost experimentu.

V databázi modelu jsou uložené následující tabulky:

- *external_suppliers* (externí dodavatelé): Jména a adresy měst externích dodavatelů.
- *production_lines* (výrobní linky): Jména a adresy měst výrobních linek dodavatelského řetězce.
- *inputs* (vstupy): Volitelně je možné využít informace o propojení vstupů v rámci dodavatelského řetězce. V modelu je propojení generováno pseudonáhodně.

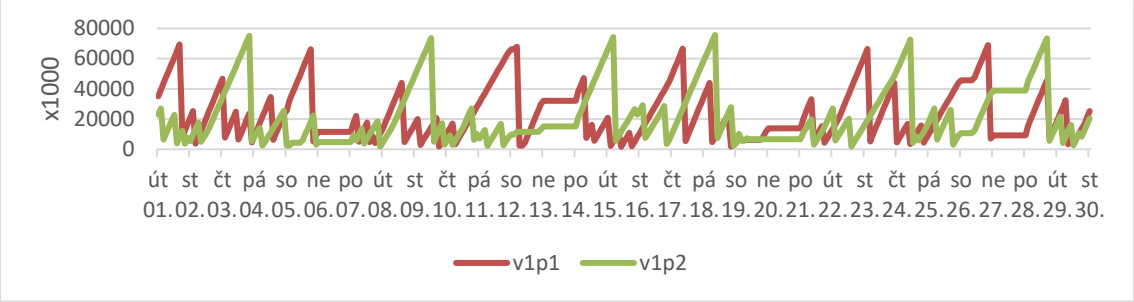
5 Experimenty

Cílem experimentů je simulace výpadků dodávek surovin a následné ověření šíření chyby v dodavatelském řetězci. Simulaci je možné spustit s různými variantami nastavení modelovaného dodavatelského řetězce. V této kapitole jsou ověřeny tři možnosti nastavení. Pro zajištění reprodukovatelnosti a shodných podmínek pro všechny tři experimenty jsou ponechány u všech parametrů modelu výchozí hodnoty. Mění se pouze varianty simulace poruchy. Výsledky simulace jsou zpracované do grafů závislosti množství produktů ve výstupních frontách jednotlivých výrobních linek na času. Svislé osy grafů představují množství produktů a vodorovné osy představují čas. Z důvodu větší přehlednosti a z důvodu velkých rozdílů mezi produktivitou jednotlivých vrstev modelu byl vytvořen graf pro každou vrstvu zvlášť. Popisky svislých os grafů prvních dvou vrstev je nutné násobit jedním tisícem. Počátek modelového času je 1. 7. 2020. Po spuštění běží simulace celkem 3 měsíce, z toho první dva měsíce se čeká na ustálení činnosti modelu. Výsledky jsou zpracovány pouze za poslední měsíc, tedy od 1. 9. 2020 do 30. 9. 2020. Pro větší přehlednost je v grafech ponechána v popiscích vodorovných os pouze zkratka názvu dne a pořadové číslo dne v měsíci. Název dne je důležitý pro ověření produktivity v pracovní dny a pro ověření zavedení pracovní soboty na lince. Pořadové číslo dne slouží především pro posouzení zpoždění šíření chyby po jednotlivých vrstvách modelu. U všech experimentů byla použita stejná konfigurace dodavatelského řetězce, viz tabulka v příloze č. 1. Tabulka obsahuje seznam vstupů jednotlivých výrobních linek, počáteční množství surovin ve vstupních frontách a počty surovin nutných pro výrobu jednoho produktu.

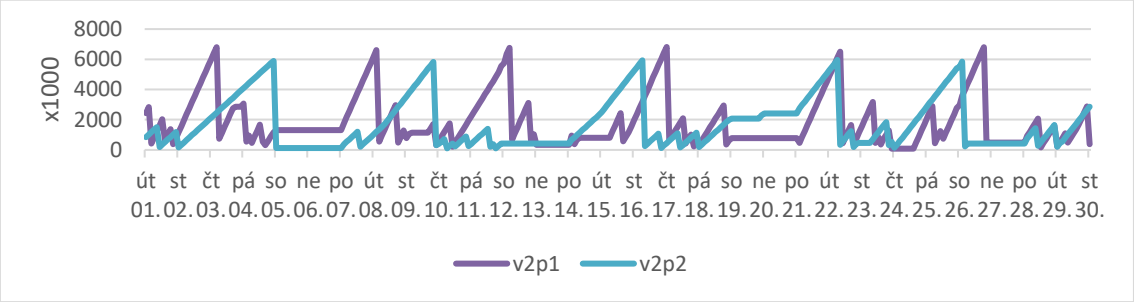
5.1 Simulace bez poruchy

Experiment č. 1 je proveden v režimu bezporuchové simulace a slouží pro porovnání s experimenty ovlivněnými poruchou a posouzení funkčnosti modelu, viz Obr. 14 – Obr. 18. Z grafů jsou patrné přestávky ve výrobě zejména v sobotu a v neděli, ale u některých linek ve 2. – 5. vrstvě lze pozorovat přerušení výroby i v pracovním týdnu, přestože agentů *truck* je v modelu dostatek. Toto chování může

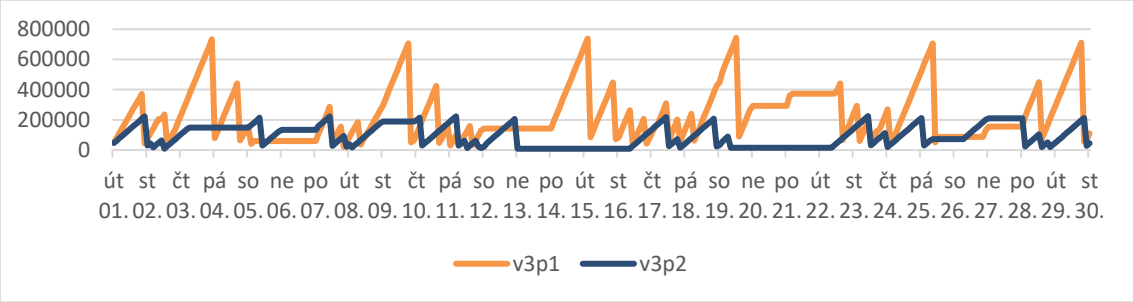
být způsobené nedostatečným zásobováním surovinami, protože ve výstupním skladu je podle grafů dostatečná volná kapacita. Mezi další možnosti patří příliš vysoká produktivita, nedostatečná poptávka, nebo kombinace všech těchto vlivů.



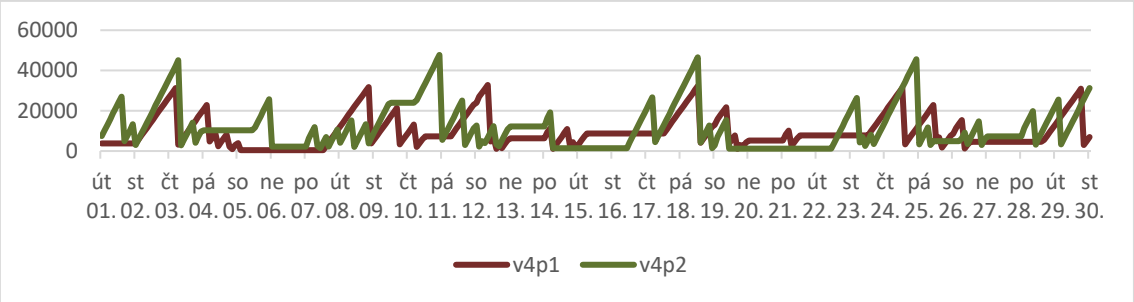
Obr. 14: Graf simulace bez poruchy (1. vrstva). Zdroj: vlastní zpracování.



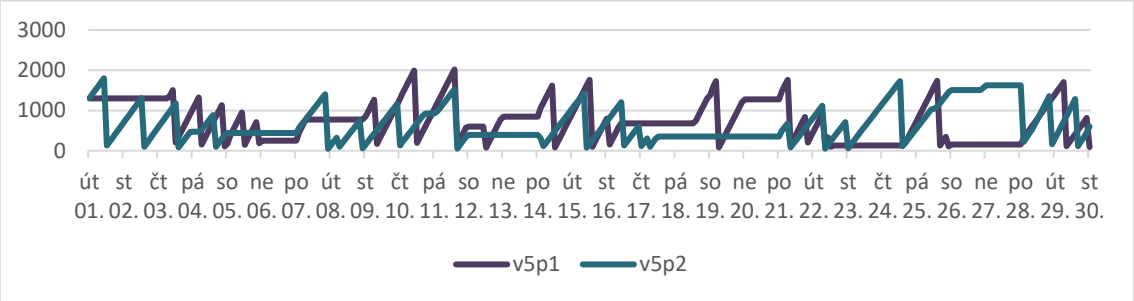
Obr. 15: Graf simulace bez poruchy (2. vrstva). Zdroj: vlastní zpracování.



Obr. 16: Graf simulace bez poruchy (3. vrstva). Zdroj: vlastní zpracování.



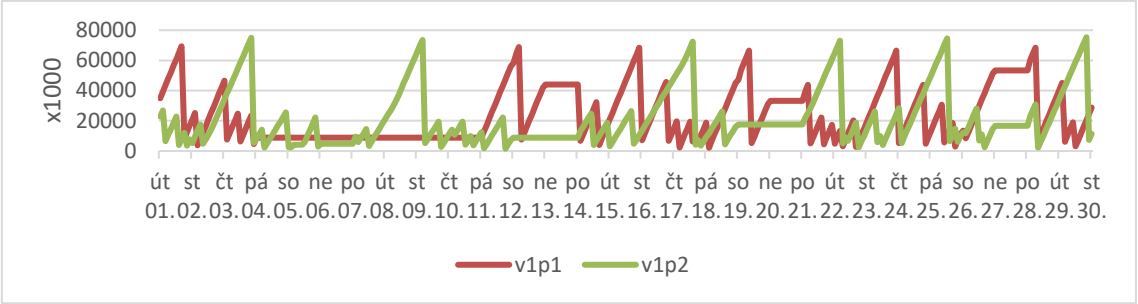
Obr. 17: Graf simulace bez poruchy (4. vrstva). Zdroj: vlastní zpracování.



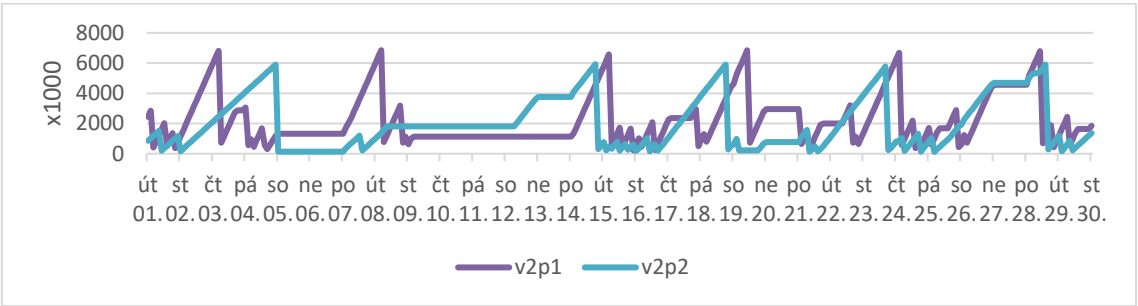
Obr. 18: Graf simulace bez poruchy (5. vrstva). Zdroj: vlastní zpracování.

5.2 Simulace poruchy bez opravy

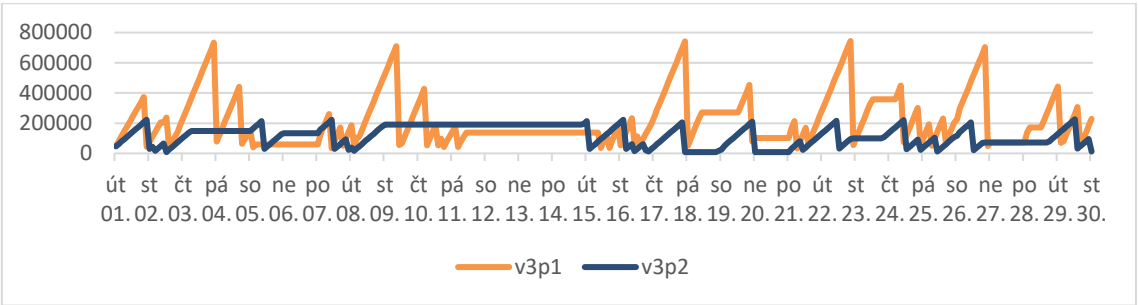
Experiment č. 2 do simulace zahrnuje poruchu výrobní linky V_1P_1, která se projeví v pátek 4. 9. a trvá do pátku 11. 9, viz Obr. 19 – Obr. 23. Tato porucha se projeví ve druhé vrstvě s cca čtyřdenním zpožděním a trvá přibližně 3,5 pracovních dní. Ve třetí vrstvě lze pozorovat chování velmi podobné jako v druhé vrstvě, více je poruchou ovlivněna linka V_3P_2. Ve čtvrté vrstvě se porucha projeví až 9. 9. a 10. 9 a výpadek výroby zde trvá cca 3 pracovní dny. Pátou vrstvu porucha ovlivňuje přibližně od 11. 9. do 16. 9. a výpadek zde trvá přibližně 3 pracovní dny. Jediná linka, která nebyla poruchou ovlivněna je V_1P_2. To je způsobené tím, že je tato linka ve stejné vrstvě jako linka s poruchou, takže není závislá na produktech porouchané linky. U simulace poruchy bez opravy lze pozorovat postupné šíření chyby dodavatelským řetězcem, přičemž v každé další vrstvě se chyba projevuje s určitým zpožděním v řádu dní, doba výpadku se postupně zkracuje.



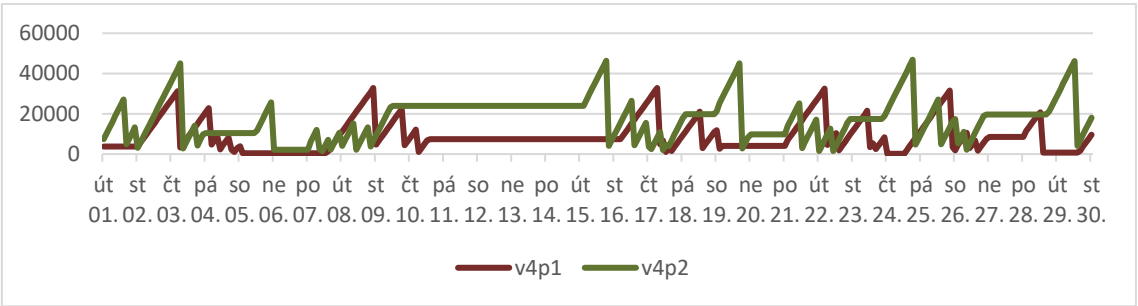
Obr. 19: Graf simulace poruchy bez opravy(1. vrstva). Zdroj: vlastní zpracování.



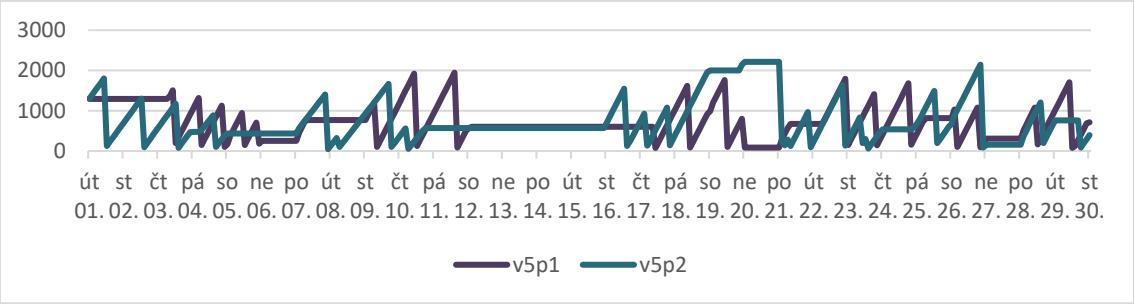
Obr. 20: Graf simulace poruchy bez opravy(2. vrstva). Zdroj: vlastní zpracování.



Obr. 21: Graf simulace poruchy bez opravy(3. vrstva). Zdroj: vlastní zpracování.



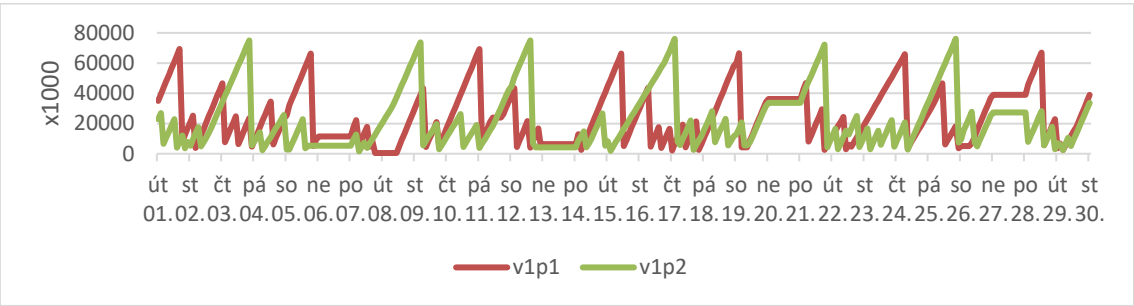
Obr. 22: Graf simulace poruchy bez opravy(4. vrstva). Zdroj: vlastní zpracování.



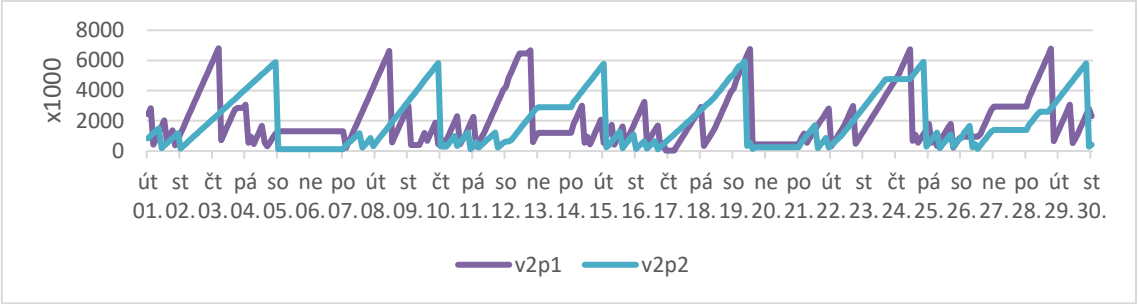
Obr. 23: Graf simulace poruchy bez opravy(5. vrstva). Zdroj: vlastní zpracování.

5.3 Simulace poruchy s opravou

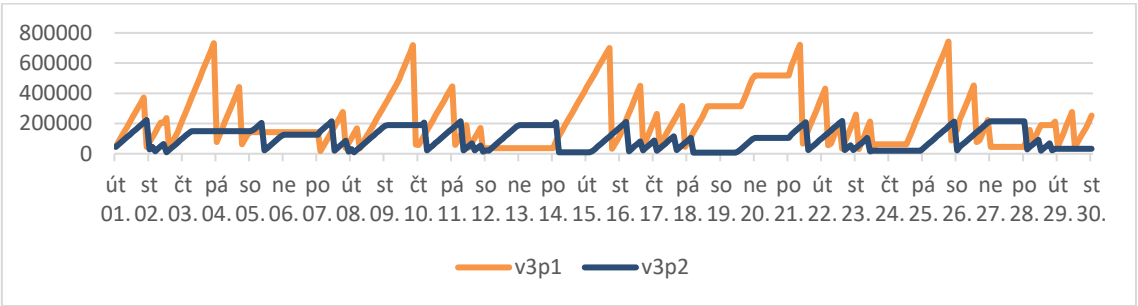
Experiment č. 3 v průběhu simulace poruchy nastavuje zahraničního externího dodavatele *ext10* (Düsseldorf) u jednoho vstupu linky V_1P_1 vybraného funkcí *uniform_discr*. Původní nastavení vstupu je po dobu trvání poruchy zálohováno do proměnných *faultInputNum* a *faultInputPL* agentu *main*. Počátek poruchy je naplánován na 4. 9. V tu chvíli je ve vstupní frontě ještě dostatek surovin, takže k objednávce dopravy dodavatelem *ext10* dojde až 7. 9. Doprava ze zahraničí trvá 34 hodin a 38 minut, náklad je vyložen 8. 9. v 10:38. Výpadek dodávky surovin trvá necelý den. Před ukončením poruchy je provedena ještě jedna objednávka ze zahraničí, přičemž výpadek trvá pouze několik hodin. Oba výpadky dodávky surovin se projeví pouze na produktivitě linky V_1P_1, produktivita ostatních linek není díky krátkému trvání poruchy ovlivněna vůbec, viz Obr. 24 – Obr. 28.



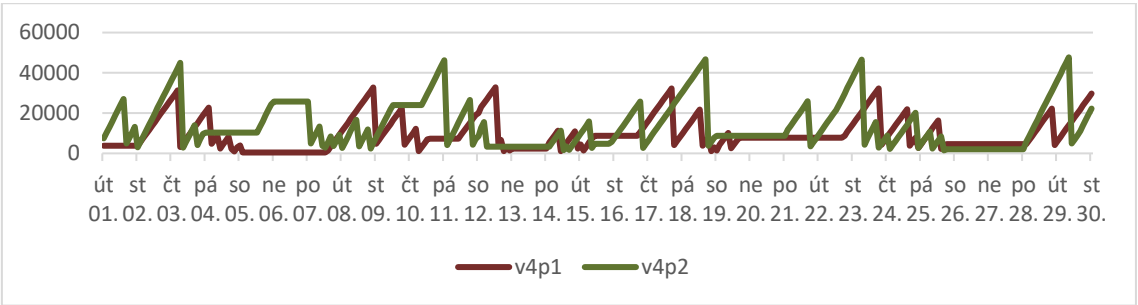
Obr. 24: Graf simulace poruchy s opravou(1. vrstva). Zdroj: vlastní zpracování.



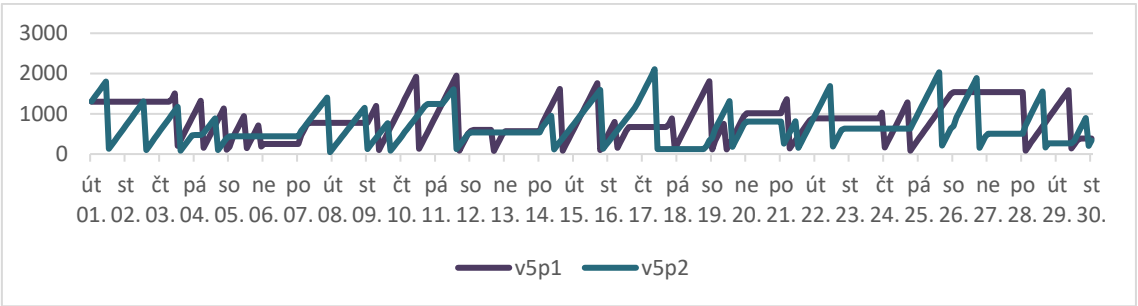
Obr. 25: Graf simulace poruchy s opravou(2. vrstva). Zdroj: vlastní zpracování.



Obr. 26: Graf simulace poruchy s opravou(3. vrstva). Zdroj: vlastní zpracování.



Obr. 27: Graf simulace poruchy s opravou(4. vrstva). Zdroj: vlastní zpracování.



Obr. 28: Graf simulace poruchy s opravou(5. vrstva). Zdroj: vlastní zpracování.

6 Závěry a doporučení

Multiagentový přístup se v průběhu vytváření a ladění modelu ukázal jako vhodné řešení, protože umožňuje v krátkém čase vytvořit model reálného světa a ověřit na něm možnosti nových metod a postupů, dřív, než budou zavedeny do skutečné výroby. Jako nevýhoda se naopak ukázalo, že i drobná chyba algoritmu, která se ani nemusí projevit u všech agentů stejného typu, dokáže zapříčinit nefunkčnost celého modelu. Takto se v praxi projevuje emergence. Model je budován zdola nahoru, takže drobná chyba na mikroúrovni může mít fatální následky na makroúrovni. To je hlavní důvod, proč vyzkoušet funkčnost nejprve na modelu. Další výhodou modelování je možnost měnit rychlost běhu modelu. Zrychlením běhu lze odhalit chyby, které by se v reálném světě projevily v budoucnosti. Pokud je nalezena chyba, ale rychlost běhu modelu je příliš vysoká, lze snížením rychlosti simulace a logováním jednotlivých operací provést ladění a zjistit přesnou příčinu chyby.

Při experimentech bylo pozorováno, že produktivitu dodavatelského řetězce z globálního pohledu ovlivňuje především doba trvání poruchy. Při včasné reakci na výskyt poruchy lze úplně zabránit šíření dopadů poruchy do výkonosti výrobních linek ve vyšších vrstvách dodavatelského řetězce. Dále bylo v rámci simulace poruchy s opravou (viz kapitola 5.3) zjištěno, že prodloužení doby dopravy surovin od rezervního dodavatele vůči době dopravy od standardního dodavatele může ovlivnit produktivitu linky, u které došlo k poruše, ale nemusí mít vliv na produktivitu ostatních linek v dodavatelském řetězci. Následky chyby jsou tedy daleko menší, než v případě simulace poruchy bez opravy (viz kapitola 5.2). Dobu výpadku linky může ovlivnit vzdálenost externího dodavatele. V případě větší vzdálenosti je možné nastavit limitní množství určující okamžik objednání surovin na vyšší hodnotu, aby byla doprava zahájena včas a výpadek byl co nejkratší.

Model je připravený pro simulaci existujícího výrobního řetězce. Z databáze je možné importovat informace o vstupech výrobních linek, kapacitách vstupních a výstupních front, množství produktů vyrobených za hodinu atd. Pro další výzkum by bylo vhodné rozšířit model dodavatelského řetězce horizontálně i vertikálně, tj. navýšit počet vrstev i výrobních linek v jednotlivých vrstvách. Dále by bylo možné

se v modelu více zaměřit na logistiku. Při objednávání dopravy lze řešit například objem, rozměry a hmotnost produktů (surovin) ve vztahu k nosnosti a rozměrům palet. Dále lze v modelu zohlednit počet a hmotnost palet, které lze naložit do návěsů a přívěsů kamionů, spotřebu pohonných hmot a případně i alternativní způsoby dopravy. Mezi další aspekty zásadně ovlivňující ekonomiku dodavatelských řetězců patří velikost skladovacích prostorů, počet zaměstnanců, atd. Výsledkem dalšího výzkumu by měl být model umožňující vytvoření přesné ekonomické analýzy, která by nalezla uplatnění při posuzování změn v reálném dodavatelském řetězci.

7 Seznam použité literatury

BLESING, Christian, Dennis LUENSCH, Jonas STENZEL a Benjamin KORTH, 2017. Concept of a Multi-agent Based Decentralized Production System for the Automotive Industry. In: Yves DEMAZEAU, Paul DAVIDSSON, Javier BAJO a Zita VALE, ed. *Advances in Practical Applications of Cyber-Physical Multi-Agent Systems: The PAAMS Collection*. Cham: Springer International Publishing, s. 19–30. ISBN 978-3-319-59930-4.

BRANISSO, Lucas Binhardi, Edilson R R KATO, Emerson Carlos PEDRINO, Orides MORANDIN a Roberto H TSUNAKI, 2013. A Multi-Agent System Using Fuzzy Logic to Increase AGV Fleet Performance in Warehouses. *2013 III Brazilian Symposium on Computing Systems Engineering*. 137–142.

CARLSSON, Christer a Robert FULLER, 2000. Soft computing and the bullwhip effect. *Econ. Complex.* **2**, 1–26.

CHAIB-DRAA, Brahim a Jörg MÜLLER, 2006. *Multiagent based Supply Chain Management* [online]. Dostupné z: doi:10.1007/978-3-540-33876-5

CHEN, Y. X. a Y. SONG, 2014. Emergency response capability assessment of emergency supply chain coordination mechanism based on hesitant fuzzy information. *International Journal of Simulation Modelling* [online]. **13**(4), 485–496. ISSN 17264529. Dostupné z: doi:10.2507/IJSIMM13(4)CO18

CUPEK, Rafal, Adam ZIEBINSKI, Lukasz HUCZALA a Huseyin ERDOGAN, 2016. Agent-based manufacturing execution systems for short-series production scheduling. *Computers in Industry* [online]. **82**, 245–258. ISSN 0166-3615. Dostupné z: doi:https://doi.org/10.1016/j.compind.2016.07.009

FIPA, 2019. FIPA. *The Foundation for Intelligent Physical Agents* [online] [vid. 2020-03-02]. Dostupné z: http://www.fipa.org/index.html

GUO, Yanli, Jianbin CHEN, Hailing GUO a Xinman LU, 2013. Coordination mechanism of SaaS service supply chain: Based on compensation contracts. *Journal of Industrial Engineering and Management* [online]. **6**. Dostupné z: doi:10.3926/jiem.789

HORLING, Bryan a Victor LESSER, 2004. A Survey of Multi-agent Organizational Paradigms. *The Knowledge Engineering Review* [online]. **19**, 281–316. ISSN 0269-8889. Dostupné z: doi:10.1017/S0269888905000317

HSIEH, Fu-Shiung, 2019. Dynamic configuration and collaborative scheduling in supply chains based on scalable multi-agent architecture. *Journal of Industrial Engineering International* [online]. **15**(2), 249–269. ISSN 2251-712X. Dostupné z: doi:10.1007/s40092-018-0291-5

IBM, 2015. CPLEX Optimizer. *High-performance mathematical programming solver for linear programming, mixed-integer programming and quadratic programming* [online] [vid. 2020-04-03]. Dostupné z: https://www.ibm.com/analytics/cplex-optimizer

KAMAGAEW, Andreas, Jonas STENZEL, Andreas NETTSTRÄTER a Michael HOMPEL, 2011. Concept of Cellular Transport Systems in facility logistics. In: *ICARA 2011 - Proceedings of the 5th International Conference on Automation, Robotics and Applications* [online]. s. 40–45. Dostupné z: doi:10.1109/ICARA.2011.6144853

KAMIŃSKI, Bogumił a Przemysław SZUFEL, 2013. Verification of Models in Agent Based Computational Economics - Lessons from Software Engineering. In: Andrzej KOBYLIŃSKI a Andrzej SOBCZAK, ed. *Perspectives in Business Informatics Research*. Berlin, Heidelberg: Springer Berlin Heidelberg, s. 185–199. ISBN 978-3-642-40823-6.

KLEIN, Matthias, Andreas LÖCKLIN, Nasser JAZDI a Michael WEYRICH, 2018. A negotiation based approach for agent based production scheduling. *Procedia Manufacturing* [online]. **17**, 334–341. ISSN 2351-9789. Dostupné z: doi:https://doi.org/10.1016/j.promfg.2018.10.054

KUBÍK, Aleš, 2004. *Inteligentní agenty*. 1. vyd. Brno: Computer Press. ISBN 80-251-0323-4.

LI, Zhendong, Huiying ZHANG, Ke JIANG a Mitchell MAINSTONE, 2019. The asymmetric game of production technology in a manufacturing supply chain network: the influence of number of manufacturing partners. *The International Journal of Advanced Manufacturing Technology* [online]. **100**(1), 797–812. ISSN 1433-3015. Dostupné z: doi:10.1007/s00170-018-2775-2

LONG, Qingqi, 2015. Three-dimensional-flow model of agent-based computational experiment for complex supply network evolution. *Expert Systems with Applications* [online]. B.m.: Elsevier Ltd, **42**(5), 2525–2537. ISSN 09574174. Dostupné z: doi:10.1016/j.eswa.2014.10.036

LU, Zhiyao, Zilong ZHUANG, Zizhao HUANG a Wei QIN, 2019. A Framework of Multi-Agent Based Intelligent Production Logistics System. *Procedia CIRP* [online]. **83**, 557–562. ISSN 2212-8271. Dostupné z: doi:https://doi.org/10.1016/j.procir.2019.04.116

LUO, Hao, Kai WANG, Xiang T R KONG, Shaoping LU a Ting QU, 2017. Synchronized production and logistics via ubiquitous computing technology. *Robotics and Computer-Integrated Manufacturing* [online]. **45**, 99–115. ISSN 0736-5845. Dostupné z: doi:https://doi.org/10.1016/j.rcim.2016.01.008

MONOSTORI, László, 2014. Cyber-physical production systems: Roots, expectations and R&D challenges. *Procedia CIRP* [online]. B.m.: Elsevier B.V., **17**, 9–13. ISSN 22128271. Dostupné z: doi:10.1016/j.procir.2014.03.115

ORMEROD, Paul a Bridget ROSEWELL, 2009. Validation and Verification of Agent-Based Models in the Social Sciences. In: Flaminio SQUAZZONI, ed. *Epistemological Aspects of Computer Simulation in the Social Sciences*. Berlin, Heidelberg: Springer Berlin Heidelberg, s. 130–140. ISBN 978-3-642-01109-2.

PARUNAK, H Van Dyke, 1996. *Applications of Distributed Artificial Intelligence in Industry*. 1996.

PLINERE, Darya a Ludmila ALEKSEJEVA, 2019. Production scheduling in agent-based supply chain for manufacturing efficiency improvement. *Procedia Computer Science* [online]. **149**, 36–43. ISSN 1877-0509. Dostupné z: doi:https://doi.org/10.1016/j.procs.2019.01.104

PLINERE, Darya a Arkady BORISOV, 2015. Case Study on Inventory Management Improvement. *Information Technology and Management Science* [online]. **18**. Dostupné z: doi:10.1515/itms-2015-0014

PRASAD, T V S R K, Kolla SRINIVAS a C SRINIVAS, 2017. Decentralized production--distribution planning in multi-echelon supply chain network using intelligent agents. *OPSEARCH* [online]. **54**(2), 217–232. ISSN 0975-0320. Dostupné z: doi:10.1007/s12597-016-0277-2

RUSSELL, S a P NORVIG, 2016. *Artificial Intelligence: A Modern Approach* [online]. B.m.: Pearson. Always learning. ISBN 9781292153964. Dostupné z: https://books.google.cz/books?id=XS9CjwEACAAJ

TANAJURA, Ana Paula M, Valdir Leanderson C OLIVEIRA a Herman LEPIKSON, 2015. A Multi-agent Approach for Production Management. In: Mitsuo GEN, Kuinam J KIM, Xiaoxia HUANG a Yabe HIROSHI, ed. *Industrial Engineering, Management Science and Applications 2015*. Berlin, Heidelberg: Springer Berlin Heidelberg, s. 65–75. ISBN 978-3-662-47200-2.

TAURINO, Teresa a Agostino VILLA, 2017. Multi-agent Systems for Production Management in Collaborative Manufacturing. In: Luis M CAMARINHA-MATOS, Hamideh AFSARMANESH a Rosanna FORNASIERO, ed. *Collaboration in a Data-Rich World*. Cham: Springer International Publishing, s. 175–182. ISBN 978-3-319-65151-4.

TER MORS, A. W., 2011. Conflict-free route planning in dynamic environments. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* [online]. s. 2166–2171. ISBN VO -. Dostupné z: doi:10.1109/iros.2011.6094461

TUCNIK, Petr, Zuzana NEMCOVA a Tomas NACHAZEL, 2017. Multiplant Production Design in Agent-Based Artificial Economic System. In: Ngoc Thanh NGUYEN, George A PAPADOPOULOS, Piotr J\KEDRZEJOWICZ, Bogdan TRAWIŃSKI a Gottfried VOSSEN, ed. *Computational Collective Intelligence*. Cham: Springer International Publishing, s. 371–380. ISBN 978-3-319-67074-4.

WOOLDRIDGE, Michael, 2009. *An Introduction to Multiagent Systems*. 2. vyd. Chichester, UK: Wiley. ISBN 978-0-470-51946-2.

ZHANG, Zhifeng a Janet DAVID, 2019. An entropy-based approach for assessing the operation of production logistics. *Expert Systems with Applications* [online]. **119**, 118–127. ISSN 0957-4174. Dostupné z: doi:https://doi.org/10.1016/j.eswa.2018.10.044

ZHAO, Jinglin a Hong ZHAO, 2018. Design of prototype system for multi-agent supply chain information sharing benefit distribution management. *Information Systems and e-Business Management* [online]. ISSN 1617-9854. Dostupné z: doi:10.1007/s10257-018-0386-y

8 Přílohy

1) Konfigurace dodavatelského řetězce
2) Zdrojový kód funkce *generateSupplyChain*
3) Zdrojový kód funkcí *addInput*, *sendTrucks*, *initFault* a *navigate*
4) Zdrojový kód událostí *startFault* a *stopFault* a funkce *setProduction*
5) Zdrojový kód funkce *runProduction*
6) Zdrojový kód funkce *initParameters* a plánovačů *weekSch* a *saturdaySch*
7) Zdrojový kód události *generateDemand*
8) CD obsahující model dodavatelského řetězce + AnyLogic 8.6.0 PLE

Konfigurace dodavatelského řetězce

linka	vstup	počet	počet / produkt		linka	vstup	počet	počet / produkt
v5p1	v4p1	18000	6		v5p2	v4p2	27000	9
	v1p2	15000	5			v3p1	21000	7
	v4p2	15000	5			v4p1	12000	4
	ext3	6000	2			v2p1	21000	7
	ext6	24000	8			ext7	6000	2
						ext3	9000	3
						ext8	6000	2
v4p1	v3p1	252000	6		v4p2	v3p2	120000	2
	v1p2	378000	9			v1p2	60000	1
	v3p2	126000	3			v3p1	420000	7
	v1p1	420000	10			v2p2	60000	1
	ext2	336000	8			v2p1	540000	9
	ext0	210000	5			ext0	300000	5
	ext4	42000	1			ext3	60000	1
						ext1	240000	4
v3p1	v2p1	3888000	4		v3p2	v2p2	3450000	10
	v1p2	3888000	4			v2p1	1725000	5
	ext9	2916000	3			ext1	1380000	4
	ext5	972000	1			ext2	690000	2
	ext3	972000	1			ext4	3450000	10
v2p1	v1p1	25938000	3		v2p2	v1p2	14742000	3
	v1p2	43230000	5			v1p1	39312000	8
	ext4	51876000	6			ext6	9828000	2
	ext2	43230000	5			ext8	4914000	1
v1p1	ext0	643566000	7		v1p2	ext5	523440000	6
	ext1	367752000	4			ext6	872400000	10
	ext2	827442000	9			ext7	697920000	8
	ext3	735504000	8			ext8	697920000	8
	ext4	735504000	8			ext9	785160000	9

Zdrojový kód funkce *generateSupplyChain*

```
ProductionLine destPL;
ProductionLine srcPL;
int mod, border, borderIn;
Input in;
ArrayList<Integer> list = new ArrayList<Integer>();
//v5p2 ... v2p1 -> 2-5x input PL + 2-3x input ext
for (int i = productionLines.size() - 1; i > 1; i--) {
	destPL = productionLines.get(i);
	traceln(destPL.name);
	if (i >= productionLines.size() - 2) destPL.baseProductivity = 1;
	else destPL.overProduction = overProduction;
	destPL.initParameters();
	//1x fix input (destPL - 2)
	srcPL = productionLines.get(i - 2);
	addInput(srcPL, destPL);
	//1-4x random input PL
	list.clear();
	mod = (i % 2);
	list.add(mod == 0 ? i - 1 : i - 3);
	border = i - mod;
	if (i > 3) {
		for (int j = 0; j < border - 2; j++) list.add(j);
		shuffle(list);
	}
	borderIn = min(4, border - 1);
	for (int j = 0; j < uniform_discr(1, borderIn); j++) {
		srcPL = productionLines.get(list.get(j));
		addInput(srcPL, destPL);
	}
	//2-3x input ext
	list.clear();
	for (int j = 0; j < externalSuppliers.size() - 1; j++) list.add(j);
	shuffle(list);
	for (int j = 0; j < uniform_discr(2, 3); j++) {
		srcPL = externalSuppliers.get(list.get(j));
		addInput(srcPL, destPL);
	}
}
//v1p1, v1p2 -> 5x ext
for (int i = 0; i < 2; i++) {
	destPL = productionLines.get(i);
	traceln(destPL.name);
	destPL.overProduction = overProduction;
	destPL.initParameters();
	for (int j = 0; j < 5; j++) {
		srcPL = externalSuppliers.get(j + i * 5);
		addInput(srcPL, destPL);
	}
}
```

Zdrojový kód funkce *addInput*

```
int numberPerProduct;
Input in;
numberPerProduct = uniform_discr(1, 10);

in = new Input(srcPL, destPL.inQueueInit * numberPerProduct,
numberPerProduct);
destPL.inputs.add(in);
if (srcPL.name.startsWith("v")) srcPL.baseProductivity +=
destPL.baseProductivity * numberPerProduct;
traceln(in);
```

Zdrojový kód funkce *sendTrucks*

```
List freeTrucks = filter(trucks, t -> t.inState(Truck.at_depot));
Iterator<Truck> truckIterator = freeTrucks.iterator();
while (!transportOrders.isEmpty() && truckIterator.hasNext()) {
	TransportOrder transportOrder = transportOrders.removeFirst();
	send(transportOrder, truckIterator.next());
}
```

Zdrojový kód funkce *initFault*

```
if (faultSimulation > 0) {
	startFault.restartTo(faultDay, DAY);
	stopFault.restartTo(faultDay + faultDuration, DAY);
} else {
	startFault.suspend();
	stopFault.suspend();
}
```

Zdrojový kód funkce *navigate*

```
selectedViewArea = viewArea;
viewArea.navigateTo();
groupMainMenu.setPos(viewArea.getX(), viewArea.getY());
```

Zdrojový kód události *startFault*

```
ProductionLine fpl = productionLines.get(faultLine);
traceln(date() + " Počátek chyby linky " + fpl.name);
if (faultSimulation == 2) {
	faultInputNum = uniform_discr(fpl.inputs.size() - 1);
	faultInputPL = fpl.inputs.get(faultInputNum).productionLine;
	fpl.inputs.get(faultInputNum).productionLine =
externalSuppliers.get(10);
} else {
	fpl.fault = true;
}
```

Zdrojový kód události *stopFault*

```
ProductionLine fpl = productionLines.get(faultLine);
traceln(date() + " Konec chyby linky " + fpl.name);
if (faultSimulation == 2) {
	fpl.inputs.get(faultInputNum).productionLine = faultInputPL;
}
productionLines.get(faultLine).fault = false;
```

Zdrojový kód funkce *setProduction*

```
//Plánování produktivity příští směny podle orders, preOrders a dostupné
kapacity výstupní fronty
final double MAX_PRODUCTIVITY = 1.4;
final double STEP = 0.2;
//pomocná proměnná ph = čas potřebný na výrobu předobjednávek v hodinách
double ph = preOrders / (double) productsPerHour;
//počet objednávek orders
int orders = chainOrders.size() + customerOrders.size();
//pre = 0...(ph == 0) - 1...(ph >= 8)
double pre = max(min(ph, 8), 0) / 8.0;
//capacity = 0...(capacity == 0) - 1...(capacity == outQueueCapacity)
double capacity = (outQueueCapacity - outQueue) / (double) outQueueCapacity;
if (orders > 5) {
	if (productivity < MAX_PRODUCTIVITY) productivity += STEP;
} else if (orders > 0) {
	if (productivity < 1 - (STEP / 2)) productivity += STEP;
	else if (productivity > 1 + (STEP / 2)) productivity -= STEP;
} else if (pre > 0){
	if (productivity < pre - (STEP / 2)) productivity += STEP;
	else if (productivity > pre + (STEP / 2)) productivity -= STEP;
} else {
	productivity = capacity;
}
if (productivity > MAX_PRODUCTIVITY) productivity = MAX_PRODUCTIVITY;
else if (productivity < 0) productivity = 0;
```

Zdrojový kód funkce *runProduction*

```
ProductOrder productOrder;
TransportOrder transportOrder;
int maxProduce, stocks, produce;
//Odběratel vytvoří objednávky surovin při poklesu úrovní zásob na vstupech
for (Input in:inputs) {
	if (!in.orderShipped && in.queue <= inQueueMin * in.numberPerProduct)
{
		//zatím objednává celou kapacitu (dodané množství se nemusí
vejít)
		productOrder = new ProductOrder(this, in, inQueueCapacity *
in.numberPerProduct);
		send(productOrder, in.productionLine);
		in.orderShipped = true;
	}
}
//Dodavatel vytvoří objednávky dopravy v rámci výrobního řetězce při dostatku
produktů na výstupu
//Produkty jsou od objednání dopravy do naložení na kamion rezervovány
boolean nextTry = true;
Iterator i = chainOrders.iterator();
while (nextTry && i.hasNext()) {
	productOrder = (ProductOrder) i.next();
	if (outQueue - reservation >= productOrder.quantity ||
name.startsWith("ext")) {
		reservation += productOrder.quantity;
		transportOrder = new TransportOrder(this, (ProductionLine)
productOrder.customer, productOrder.input, productOrder.quantity);
		main.transportOrders.addLast(transportOrder);
		preOrders -= productOrder.quantity;
		i.remove();
	} else nextTry = false;
}
//Dodavatel předá zboží zákazníkům při dostatku produktů na výstupu
nextTry = true;
i = customerOrders.iterator();
while (nextTry && chainOrders.size() == 0 && i.hasNext()) {
	productOrder = (ProductOrder) i.next();
	if (outQueue - reservation >= productOrder.quantity) {
		outQueue -= productOrder.quantity;
		//preOrders -= productOrder.quantity;
		i.remove();
	} else nextTry = false;
}
main.sendTrucks();
//Výroba, výstupní fronta (stav < max) & vstupní fronty (stav > 0)
if (outQueue < outQueueCapacity) {
	maxProduce = outQueueCapacity - outQueue;
	for (Input in:inputs) {
		stocks = (int) floor(in.queue / in.numberPerProduct);
		if(maxProduce > stocks) maxProduce = stocks;
	}
```

```
	if (maxProduce > 0) {
		//Nastavení množství výrobků za hodinu podle aktuální
produktivity
		if (productivity == 0) productivity = 0.6;
		produce = (int) ceil(productsPerHour * productivity);
		if (produce > maxProduce) produce = maxProduce;
		for (Input in:inputs) {
			in.queue -= produce * in.numberPerProduct;
		}
		outQueue += produce;
	}
}
```

Zdrojový kód funkce *initParameters*

```
baseProductivity = (int) ceil(baseProductivity * overProduction);
inQueueCapacity = main.inQueueCapacity * baseProductivity;
inQueueMin = main.inQueueMin * baseProductivity;
inQueueInit = main.inQueueInit * baseProductivity;
outQueueCapacity = main.outQueueCapacity * baseProductivity;
productsPerHour = main.productsPerHour * baseProductivity;
```

Zdrojový kód plánovače *weekSch*

```
if (name.startsWith("v")) {
	setProduction();
}
if (!fault) {
	runProduction();
}
```

Zdrojový kod plánovače *saturdaySch*

```
if (name.startsWith("v")) {
	setProduction();
}
if (!fault && productivity > 1) {
	runProduction();
}
```

Zdrojový kód události *generateDemand*

```
int pl;
ProductionLine from;
int order;
ProductOrder productOrder;
pl = uniform_discr(0, main.productionLines.size() - 1);
from = main.productionLines.get(pl);
if (pl >= main.productionLines.size() - 2) {
      order = (int) triangular(from.productsPerHour,
min(from.productsPerHour * 48, from.outQueueCapacity));
} else {
      order = (int) triangular(from.productsPerHour,
min(from.productsPerHour * 24, from.outQueueCapacity));
}
productOrder = new ProductOrder(this, null, order);
send(productOrder, from);
```

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor: **Aleš Lajvr**

Studium: I1700295

Studijní program: B1802 Aplikovaná informatika

Studijní obor: Aplikovaná informatika

Název bakalářské práce: **Multiagentový přístup k řešení logistických problémů víceinstanční průmyslové výroby**

Název bakalářské práce AJ: Multi-Agent Approach to Logistic Problems in Multi-Plant Production Logistics

Cíl, metody, literatura, předpoklady:

Práce je zaměřena na řešení vnitropodnikové logistiky podniku s více výrobními závody. Předpokládá se několikastupňová výroba, kdy výstupy prvních fází zpracování tvoří vstupy do dalších etap výroby. Výrobní a přepravní kapacity podniku budou vhodným způsobem reprezentovány agenty. Bude vytvořen zjednodušený síťový model (formou grafu) pro reprezentaci úlohy a experimentálně v něm bude ověřena funkčnost navrženého řešení pro efektivní zajištění kontinuity výroby.

Osnova práce:

1. Úvod

2. Cíle práce

3. Teoretická část a základní pojmy

4. Metodický postup řešení

5. Praktická část

6. Experimenty

7. Závěr

8. Použitá literatura

HORLING, Bryan a Victor LESSER, 2004. A survey of multi-agent organizational paradigms. *Knowledge Engineering Review,* **19**(4), 281-316. ISSN 02698889.

Russel S., Norvig. P., 2016: Artificial Intelligence: A Modern Approach (Global Ed.). Pearson Edu. ISBN 1292153962.

Wooldridge M., 2009. An Introduction to Multiagent Systems. John Wiley {\& Sons, ISBN 9780470519462.

Garantující pracoviště: Katedra informačních technologií, Fakulta informatiky a managementu

Vedoucí práce: RNDr. Petr Tučník, Ph.D.

Oponent: prof. RNDr. Peter Mikulecký, Ph.D.

Datum zadání závěrečné práce: 21.10.2014